HEWLETT · PACKARD SOFTWARE

HPL SOFTWARE LIBRARY

FOR THE 6940B

MODEL 14556A

USER'S MANUAL

HP Part No. 14556-90001
Microfiche Part No. 14556-90002

June, 1979

TABLE OF CONTENTS

Section		Page No.
I	INTRODUCTION	1-1
	GENERAL	1-1
	CALL SUBPROGRAMS	1-1
II	USING SOFTWARE LIBRARY	2-1
	GENERAL	2-1
	OPERATING INSTRUCTIONS	2-1
	MARKING TAPE FILES	2-3
	ERROR MESSAGES	2-3
III	PROGRAM DESCRIPTIONS	3-1
	UTILITY PROGRAMS	3-1
	File Lister (File 0)	3-1
	Auto Loader (File 1)	3-2
	CARD SUBPROGRAMS	3-5
	File 2 Parameter Check (PC)	3-6
	File 3 Simple Output (oo)	3-8
	File 4 Serial Output (SO)	3-10
	File 5 Parallel Output (PO)	3-12
	File 6 Interrupt Output (IO)	3-14
	File 7 Simple Input (ii)	3-17
	File 8 Serial Input (SI)	3-19
	File 9 Parallel Input (PI)	3-21
	File 10 Clear Unit (CU)	3-24
	File 11 Voltage Serial Output (VS)	3-26
	File 12 Voltage Parallel Output (VP)	3-29
	File 13 Current Serial Output (CS)	3-32
	File 14 Current Parallel Output (CP)	3-35
	File 15 Analog Input (AI)	3-38
	File 16 Single Ended Scan (SE)	3-40
	File 17 Double Ended Scan (DE)	3-44
	File 18 Null File	
	File 19 Motor Output (MO)	3-48
	File 20 Set Counter (SC)	3-50
	File 21 Read Counter (RC)	3-52
	File 22 Timer Pulse (TP)	3-54
	File 23 Relay Output (RO)	3-57
	File 24 Time Interval (TI)	3-60
	File 25 Frequency Measurement (FM)	3-63
	File 26 Option 40 Voltage Control (VC)	3-66
	File 27 Option 40 Current Control (CC)	3-70
	File 28 Arm (AM)	3-73
	File 29 Serial Poll (SP)	3-76
	File 30 Disarm (DM)	3-79
	File 31 Time Out (TO)	3-82
	File 32 Recycle (RE)	3-84
	File 33 Reference Word (RW)	3-87
	File 34 T Thermocouple (TT)	3-92
	File 35 J Thermocouple (JT)	3-95
	File 36 E Thermocouple (ET)	3-98
	File 37 R Thermocouple (RT)	3-101
	File 38 S Thermocouple (ST)	3-104
	File 39 Thermocouple Reference (TR)	3-108
	File 40 B Thermocouple (BT)	3-111
	File 41 K Thermocouple (KT)	3-115
APPENDIX A - CHANGING THE INTERFACE SELECT CODES		A-1
APPENDIX B - ADDING CALL SUBPROGRAMS		B-1

SECTION I
INTRODUCTION

GENERAL

The Hewlett-Packard program cartridge 14556A provides "friendly" programming access to the I/O functions of the 6940B Multiprogrammer System. The HP 9825A Desktop Computer is used as the controller with either the GPIO interface (Model 98032A-Option 040) or HP-IB (Model 98034A). Programs for both interfaces are recorded on the cartridge.

The 9825A must be equipped with the General I/O, Extended I/O, Advanced Programming, and String Variable ROM's in order to run the programs. When the 6940B is used with the GPIO interface, the select code is normally set to "9" When using the HP-IB interface, the select code is normally set to "7". Procedures for changing the select codes are given in Appendix A. The 6940B Multiprogrammer User's Guide (HP Part No. 59500-90005) is the general reference for system installation, operation, and programming instructions.

CALL SUBPROGRAMS

Programs on the 14556A program cartridge are accessed using a call (cll) statement followed by the name of the subprogram enclosed in single quotes. The following descriptions assume that the reader has knowledge of call subprograms. Call subprogram use is described in the Advanced Programming ROM Manual, HP Part No. 09825-90021.

Syntax

All call subprograms for the 14556A cartridge are of this type:

cll 'label' (S0, D0, S1, D1, ----, U)

Call labels can either be "long labels" or "short labels." The long labels are more descriptive (e. g., Parallel Output) while the short labels are abbreviations (e. g., PO). The long labels facilitate easier reading of the program but either label will cause the same function to be programmed. Table 1-1 lists the 14556A call subprograms in file number order with corresponding "long" and "short" program labels. Programs for the GPIO interface are recorded on track 0 while HP-IB programs are recorded on track 1 of the cartridge. Both tracks have the same file organization.

Slot Addresses

S0, S1, S2, etc. represent the card slot addresses 400, 401, 402, etc. printed beneath each plug-in card slot (see Figure 1-1). If desired, the numbers 0, 1, 2, etc. can be used instead of 400, 401, 402, etc. For example the following calls perform the same function:

cll 'Parallel Output' (401, A, 403, B)
cll 'PO' (1, A, 3, B)
cll 'Parallel Output' (401, A, 3, B)

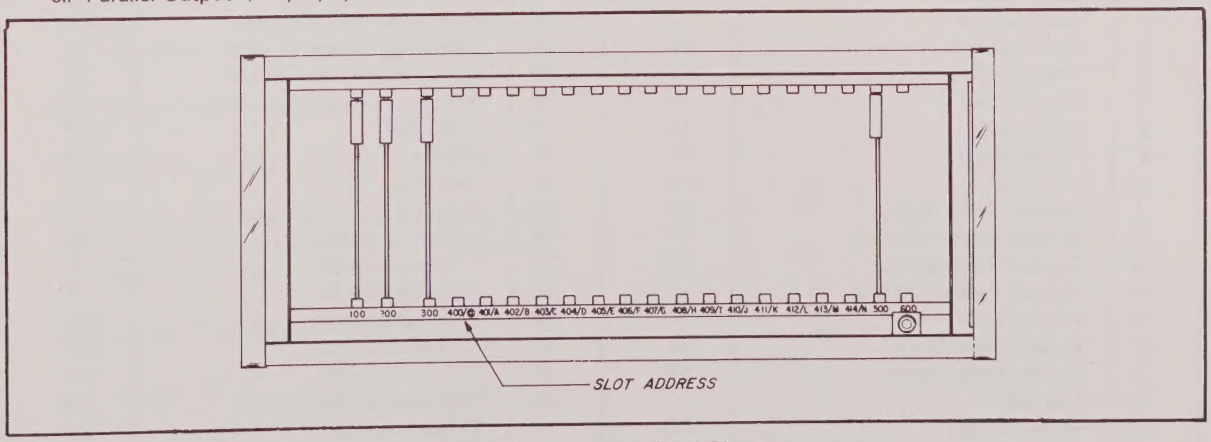


Figure 1-1. Plug-In Card Slots

Data

D0, D1, D2, etc. denote the data that is transferred between the controller and the multiprogrammer. For output transfers, D0, D1, etc. are sent from the controller to the corresponding card (S0, S1, etc.) and can represent variables or can be given as the constants. For example:

Sending Variable Data:
cll 'Parallel Output' (407, A, 402, C, 0)

Sending Constants:
cll 'Parallel Output' (407, 1234, 402, 7777, 0)

For input transfers (multiprogrammer to controller), D0, D1, etc. must be variables. The variables can be simple variables, r-variables, or single elements of any array. The returned value is placed in the proper variable. For example:

cll 'Parallel Input' (407, A, 403, r1, 405. C[3], 2)

In this example, the value from the card in slot 407 is placed in A, the value from the card in slot 403 is placed in r1, and the value from slot 405 is placed in C[3].

Unit No.

The last parameter, U, is optional. This parameter designates the unit (mainframe 0 through 15) containing the cards being called. This means that all cards called in a subroutine must be installed in the same unit. Unit 0 (the 6940B mainframe) is assigned when this parameter is omitted. For example, the following calls perform the same function:

cll 'Parallel Input' (408, A, 401, B, 0)
cll 'Parallel Input' (408, A, 401, B)

Summary

Card Slot addresses and unit numbers are always given in decimal. However, data may be either octal or decimal. Each individual Library routine specifies the proper form for data. All analog Library routines will use decimal data. For example, the Voltage Serial Output (VS) routine uses data in decimal form (e. g., −5.12 for −5.12 volts). Analog input routines(e.g., Single Ended Scan will return the voltage measured in decimal complete with polarity.

Table 1-1. 6940B Software Library for 9825A (14556A Cartridge Organization)

PROGRAM LABELS			PROGRAM LABELS		
FILE NO.	LONG	SHORT	FILE NO.	LONG	SHORT
0	File Lister	—	21	Read Counter	RC
1	Auto Loader	—	22	Timer Pulse	TP
2	Parameter Check	PC	23	Relay Output	RO
3	Simple Output	oo	24	Time Interval	TI
4	Serial Output	SO	25	Frequency Measurement	FM
5	Parallel Output	PO	26	Option 40 Voltage Control	VC
6	Interrupt Output	IO	27	Option 40 Current Control	CC
7	Simple Input	ii	28	Arm	AM
8	Serial Input	SI	29	Serial Poll	SP
9	Parallel Input	PI	30	Disarm	DM
10	Clear Unit	CU	31	Time Out	TO
11	Voltage Serial Output	VS	32	Recycle	RE
12	Voltage Parallel Output	VP	33	Reference Word	RW
13	Current Serial Output	CS	34	T Thermocouple	TT
14	Current Parallel Output	CP	35	J Thermocouple	JT
15	Analog Input	AI	36	E Thermocouple	ET
16	Single Ended Scan	SE	37	R Thermocouple	RT
17	Double Ended Scan	DE	38	S Thermocouple	ST
18	Null File	—	39	Thermocouple Reference	TR
19	Motor Output	MO	40	B Thermocouple	BT
20	Set Counter	SC	41	K Thermocouple	KT

NOTE: Programs for the GPIO interface are recorded on track 0 and the HP-IB interface programs are recorded on track 1.

SECTION II
USING THE SOFTWARE LIBRARY

GENERAL

This section describes the operating instructions for using the 14556A Software Library. Before proceeding with the operating instructions, read the general information provided in Section I and check Table 2-1 (Program Reference Table) to select the Library subprograms that can be used with your particular plug-in card configuration. Table 2-1 provides an insight into how the library works by giving a brief description of each routine and listing the plug-in cards that can be used with each call. Also, before proceeding, make a list of the plug-in card slot addresses. Once the Library routines have been selected and the slot addresses are known, the user can write his program following the instructions outlined in Figure 2-1 and described in the next paragraph.

OPERATING INSTRUCTIONS

The following steps are required to use the 14556A Software Library:

1. Write and record application program (User's program).

   a. Write the application program making use of the selected Library subprograms (see sample program below). Be sure to include an "end" statement at the bottom of the program.
   b. Enter application program into 9825A memory.
   c. Record the application program on a cartridge file. (A blank file in the 14556A Software Library could be used).

Sample Application Program

```
0: "Test Program Track 0: File 50":
1: cll 'Clear Unit'(0)
2: cll 'Serial Output'(401,37,400,7777)
3: cll 'Parallel Input'(404,A)
4: cll 'Dog'
5: cll 'SO'(409,42,0)
6: prt "Data Value in Octal";A
7: end
*18122
```

2. Insert 14556A Software Library cartridge.

   a. Select track 0 for GPIO interface or track 1 for HP-IB interface.
   b. Load the Auto Loader from file 1 of the library.
   c. Press RUN.

3. When the 9825A display shows:

   ```
   Insert User Program Tape
   ```

   a. Remove the 14556A Software Library cartridge.
   b. Insert the tape cartridge prepared in step 1c.
   c. Press CONTINUE.

4. When the 9825A display shows:

   ```
   Track # ?
   ```

Enter the track number where the program prepared in step 1 is stored. Press CONTINUE.

5. When the 9825A display shows:

```
Program FILE no. ?
```

Enter the file number where the program prepared in step 1 is stored. Press CONTINUE.

6. When the 9825A display shows:

```
Insert SOFTWARE LIBRARY
```

a. Remove the tape cartridge prepared in step 1c.
b. Insert the 14556A Software Library cartridge.
c. Press CONTINUE.

7. Check the 9825A printer.

a. If subprograms in the user program are not found in the 14556A Software Library directory, the following printout will occur:

```
Not in Directory
cll 'Dog'
```

Could be a valid user prepared subprogram label or a call label for the 14556A software library which was typed incorrectly. Note that if the sample user's program (see line no. 4) was used in step 1c, the cll 'Dog' would be printed out.

(1) If call label was not found in 14556A directory because it is a valid user prepared subprogram, refer to Appendix B for procedures explaining how to add these programs.

(2) If call label was not found because the user typed in an incorrect 14556A call label, go back to step 1 and correct your program.

b. After all required subprograms found in the directory and used in the user program are loaded into memory, the following printout will occur:

```
Subroutines from
the Directory
have been loaded
```

and the display will show:

```
Record or Edit?
```

c. At this point the user may choose to record, edit, or run the program stored in the calculator. If it is desired to record the program, proceed to step 8. To edit or run the program, go to step 9.

8. To record program, proceed as follows:

NOTE

The program can be recorded on a previously marked file on the 14556A cartridge or the user's own cartridge. Procedures for marking files are given in the next paragraph.

a. Slide cartridge tab to RECORD position and insert cartridge into 9825A.
b. Enter the word "Record" and press CONTINUE.
c. When the 9825A display shows:

```
What trk?
```

Enter the applicable track number and press CONTINUE.

d. When the 9825A display shows:

```
What File?
```

Enter the File number where the program is to be stored. Press CONTINUE.

e. When the 9825A display shows:

```
Done
```

User's completed program (Auto Loader is automatically removed) has been recorded on specified track and file. Note that the Auto Loader is still part of the program stored in calculator. If the user wishes to run the program stored in the calculator, he must first remove the Auto Loader by executing: del 0,56,*

9. To edit and/or run the program, proceed as follows:

a. Enter the word "Edit" and press CONTINUE. Printer reads:

```
To remove loader
type in and
execute:
    del 0,56,*
```

b. After the Auto Loader is removed by executing the above statement, the user can edit and/or run the program. If desired, additional user prepared subprograms can be added during the edit phase. The completed program can be stored on a previously marked file (on 14556A cartridge or user's own cartridge).

MARKING TAPE FILES

The user first determines how many bytes of program he wishes to record. This can be done by executing a "list-1" from the keyboard when the program is stored in memory. The user then inserts his tape cartridge into the 9825A and finds which file he will use to record the program into. This can be done by a "tlist" which will show which file number is the last one on the tape. The user now executes a "fdfN" where N is the last file on the tape. Now the user executes a "mrk1,B" where B is the number of bytes the user wishes to allow for. Usually, the user will wish to mark a file for more bytes than are in his program to allow for future editing.

ERROR MESSAGES

While the program is running, the slot address, card data, and unit number parameters are checked to determine if they are within limits. The Parameter Check Subprogram (File 2) checks for the following limits:

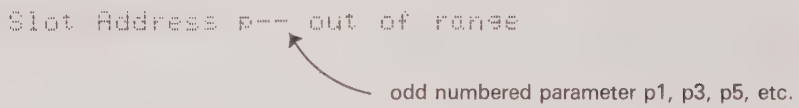
Parameter	Limits
Slot address	400 to 414, or 0 to 14
Output data	0 to 7777*
Unit number	0 to 15 (0 is assigned if this parameter is omitted)

*Certain other subprograms (e.g., Analog Output Subprograms) check output data parameters for specific decimal value limits (e.g., +10.235). The specific limits are given in Table 2-1 and in the applicable program description provided in Section III.

Parameters that are out of range will cause the program to stop and wait for a new value before continuing. The calculator display shows an "Error Message" that specifies which parameter is out of range.

Slot Address

For an out of range slot address, the display shows:



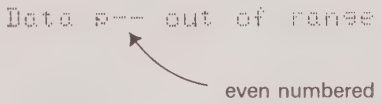
odd numbered parameter p1, p3, p5, etc.

To continue the program, type in the correct slot address and assign it to the parameter number specified in the error message. Example, if parameter p3 is out of range:

Type: 403 → p3 and press EXECUTE
Press: CONTINUE

Output Data

For out of range data, the display shows:



even numbered parameter p2, p4, p6, etc.

To continue the program, type in the correct output data value and assign it to the parameter number specified in the error message. Example, if parameter p6 is out of range:

Type: 3777 → p6 and press EXECUTE
Press: CONTINUE

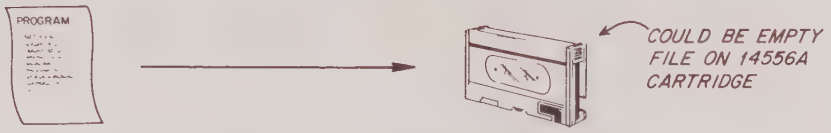
Unit Number

For an out of range unit number, the display shows:

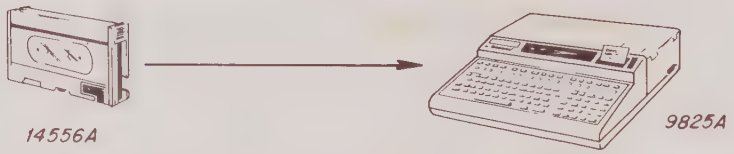
unit number > 15

To continue the program, type in the correct unit number and assign it to parameter pp0. Example:

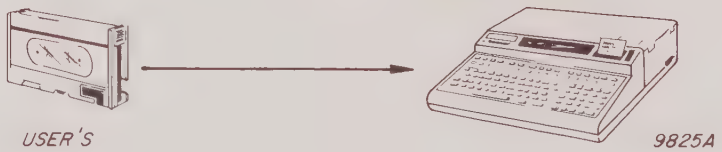
Type: 3 → pp0 and press EXECUTE
Press: CONTINUE

1. 

USER WRITES AND RECORDS HIS PROGRAM

2. 

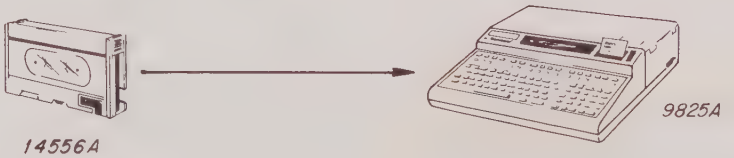
SELECT TRACK Ø (GPIO) OR TRACK 1 (HP-IB)
LOAD AUTO LOADER (FILE 1)
PRESS RUN

3. 

PRESS CONTINUE

4. ENTER TRACK NO. WHERE USER'S PROGRAM IS RECORDED.
PRESS CONTINUE

5. ENTER FILE NO. WHERE USER'S PROGRAM IS RECORDED.
PRESS CONTINUE

6. 

PRESS CONTINUE

7. EDIT, RUN, OR RECORD THE PROGRAM

NOTE: BEFORE EDITING OR RUNNING THE PROGRAM, THE AUTO LOADER MUST BE REMOVED BY EXECUTING:

del Ø, 56, *

Figure 2-1. Using 14556A Software Library

TABLE 2-1. PROGRAM REFERENCE TABLE

Type	Program Label	Plug-in Cards	File No.	Description
Digital Output	Interrupt Output (IO)	69330A Relay Output 69331A Digital Output	6	Sends octal data (0 to 7777$_8$) to the output cards specified. Uses the 9825A's interrupt capability to indicate when data transfers with external devices are completed. See page 3-14.
	Parallel Output (PO)	69330A Relay Output 69331A Digital Output 69433A Relay/Readback	5	Sends octal data (0 to 7777$_8$) to output cards specified. Card outputs are updated simultaneously. See page 3-12.
	Relay Output (RO)	69433A Relay/Readback	23	Changes the state of the desired relay (1 through 12) on the specified card. Data must be in the range of 1 to 12 or −12 to −1. Relays are closed for positive numbers and opened for negative numbers. See page 3-57.
	Serial Output (SO)	69330A Relay Output 69331A Digital Output 69433A Relay/Readback	4	Sends octal data (0 to 7777$_8$) to the output cards specified. Card outputs are updated sequentially. See page 3-10.
	Simple Output (oo)	69332A Open Collector 69380A Breadboard Output	3	Sends octal data (0 to 7777$_8$) to the output card (only one card per call) specified. Does not cause 6940B to wait for flag. See page 3-8.
Analog Output	Option 40 Current Control (CC)	69510A through 69513A Resistance Output	27	Sends desired current output (0 to 1.000 amps) in decimal form (e.g., 0.55 for 0.55A) to the option 40 power supply programming cards specified. Each card has 2 output channels. See page 3-70.
	Current Serial Output (CS)	69370A Current D/A	13	Sends desired output current (0 to 20.475mA) in decimal form (e.g., 4.5 for +4.5mA) to the cards specified. Cards are updated sequentially to their new output currents. See page 3-32.
	Current Parallel Output (CP)	69370A Current D/A	14	Sends desired output current (0 to 20.475mA) in decimal form (e.g., 4.5 for +4.5mA) to the cards specified. The cards are simultaneously updated to their new output currents. See page 3-35.
	Option 40 Voltage Control (VC)	69500A through 69506A Resistance Output	26	Sends desired output voltage (0 to 100.00) in decimal form to the option 40 power supply programming cards specified. See page 3-66.
	Voltage Parallel Output (VP)	69321B Voltage D/A 69322A Quad D/A	12	Sends desired output voltage (−10.240 to +10.235V) in decimal form (e.g., 7.5 for +7.5 volts) to the cards specified. 69322A Quad D/A card outputs are updated when the applicable outputs are addressed (e.g., to address channel 2 of 69322A in slot 3 use address 3.2). The 69321B cards are simultaneously updated to their new output voltages. See page 3-29.

TABLE 2-1. PROGRAM REFERENCE TABLE (Continued)

Type	Program Label	Plug-in Cards	File No.	Description
Analog Output (cont.)	Voltage Serial Output (VS)	69321B Voltage D/A 69322A Quad D/A	11	Sends desired output voltage (−10.240 to +10.235V) in decimal form (e.g., −6.5 for −6.5 volts) to the cards specified. Card outputs are updated sequentially. Note that a 69322A card has four output channels which can be individually updated. (e.g., to update channel 2 of 69322A in slot 3 use address 3.2). See page 3-26.
Digital Input	Parallel Input (PI)	69431A Digital Input	9	Returns octal data (IRQ bit removed) from the input cards specified. The first card that has data ready is read first, followed by the next ready card, etc. The subprogram will not complete until all cards have been read. See page 3-21.
	Serial Input (SI)	69431A Digital Input	8	Returns octal data (IRQ bit removed) from the input cards specified. Cards are read sequentially. See page 3-19.
	Simple Input (ii)	69430A Isolated Digital Input 69480A Breadboard Output	7	Reads octal data from the input card (only one card per call). The data is returned in octal (IRQ bit not removed). Does not cause the 6940B to wait for flag. See page 3-17.
Analog Input	Analog Input (AI)	69421A Voltage Monitor 69422A High Speed A/D	15	Reads voltage data (−10.240 to +10.235V) from the cards specified. Returned data is in decimal form (e.g., −5.5 for −5.5 volts). Cards are read sequentially. See page 3-38.
	Double Ended Scan (DE)	69421A Voltage Monitor 69422A High Speed A/D 69330A Relay Output 69433A Relay/Readback	17	Scans and reads voltages from multiplexed DC sources. Measured voltages in decimal form are read from 69421A or 69422A cards. The 69330A or 69433A cards are used to scan the dc voltages to be measured. The voltages scanned need not share a single common or ground wire. All readings are stored in r-variables. See page 3-44.
	Single Ended Scan (SE)	69421A Voltage Monitor 69422A High Speed A/D 69330A Relay Output 69433A Relay/Readback	16	Scans and read voltages from multiplexed dc sources. Measured voltages in decimal form are read from 69421A or 69422A cards. The 69330A or 69433A cards are used to scan the dc voltages to be measured. All readings are stored in r-variables. The voltages scanned must share a single common or ground wire. If this is not possible, see file 17, "Double Ended Scan". See page 3-41.
Pulse Counting	Read Counter (RC)	69435A Pulse Counter	21	Reads the specified counter cards. Returned data is in decimal form and is stored in the specified variables. See page 3-52.
	Set Counter (SC)	69435A Pulse Counter	20	Presets specified counters to the decimal count sent as data. Maximum count that card can be set to is 4095. See page 3-50.

TABLE 2-1. PROGRAM REFERENCE TABLE (Continued)

Type	Program Label	Plug-in Cards	File No.	Description
Stepping Motor Control	Motor Output (MO)	69335A Stepping Motor Control	19	Sends signed decimal data to the card specified. Positive numbers will cause stepping motors to rotate in one direction, and negative numbers in the reverse direction. Data greater than ±2047 steps will cause a stop. See page 3-48.
Timing	Frequency Measurement (FM)	69435A Pulse Counter 69600B Timer	25	Returns the decimal value of an external frequency source. The program adjusts the sample interval (from 69600B) until the best resolution is achieved. See page 3-63.
	Time Interval (TI)	69435A Pulse Counter 69601B Frequency Reference	24	Returns the decimal value of an external pulse. The time base (69601B) is assumed to be 100kHz. See page 3-60.
	Timer Pulse (TP)	69600B Timer	22	Sends the time (in seconds) to the card specified. Time interval resolution is 0.001 seconds. This call assumes a jumper wire has been installed between output connector (P2) pins 4 and 11. See page 3-54.
Interrupt Mode	Arm (AM)	69431A Digital Input 69434A Event Sense 69436A Process Interrupt 69600B Timer	28	Arms cards specified to permit an interrupt whenever any card detects an interrupt condition. (Assumes all W6 jumpers have been removed.) After last card is armed, program puts the Multiprogrammer system in interrupt mode and enables 9825A interrupt system before returning. See page 3-73.
	Disarm (DM)	69431A Digital Input 69434A Event Sense 69436A Process Interrupt 69600B Timer	30	Selectively disarms cards specified. Assumes all W6 jumpers have been removed. See page 3-79.
	Recycle (RE)	69436A Process Interrupt 69600B Timer	32	Recycles specified cards, before they are armed by the subprogram 'Arm' (File 28). See page 3-84.
	Reference Word (RW)	69434A Event Sense	33	Reads back the reference words from the desired Event Sense cards and sends new reference words to these cards. See page 3-87.
	Serial Poll (SP)	69431A Digital Input 69434A Event Sense 69436A Process Interrupt 69600B Timer	29	Performs a Serial Poll on cards specified. The data returned is in octal and includes an IRQ bit if the card generated an interrupt. See page 3-76.

TABLE 2-1. PROGRAM REFERENCE TABLE (Continued)

Type	Program Label	Plug-in Cards	File No.	Description
Temperature Measurement	B Thermocouple (BT)	69423A Low Level A/D	40	Reads the temperature in °C from the 69423A card and the applicable thermocouple type. A 69423A card has six input channels that are read individually. For example, to read channel 5 of the 69423A card in slot 2, use address 402.5 or 2.5. See pages 3-92 through 3-118.
	E Thermocouple (ET)		36	
	J Thermocouple (JT)		35	
	K Thermocouple (KT)		41	
	R Thermocouple (RT)		37	
	S Thermocouple (ST)		38	
	T Thermocouple (TT)		34	
	Thermocouple Reference (TR)		39	Reads the temperature in °C of the reference on the 69423A card. See page 3-108.
Special	Clear Unit (CU)	All	10	Sends logic zeroes to clear all cards in unit specified. For example, sending logic zeroes will clear a Relay Output Card (open all 12 relays). See page 3-24.
	Time Out (TO)	All Output Cards	31	Halts system until all output cards complete their operations. See page 3-82.

SECTION III
PROGRAM DESCRIPTIONS

UTILITY PROGRAMS

The following paragraphs contain detailed descriptions of the File Lister and Auto Loader utility programs which are recorded on Files 0 and 1, respectively. Program listings are included with each description.

File Lister (File 0)

The File Lister program is the same for both the GPIO and the HP-IB interfaces and can be used with any 9825A program cartridge. The program starts by loading file zero (from the current track) off the program cartridge in the tape transport. The program from the cartridge is loaded into memory after all program lines in the File Lister. After the program is in memory, the first line of that program is stored in a string variable by lines 10 and 16. The string is then printed on the printer whose select code is represented by the variable P. With P = 16, the printout occurs on the internal strip printer. Null files, binary program files, etc. will not be loaded, but will cause a printout of the file type, thus:

```
FILE # 40
 0: "B Thermocou
ple":

FILE # 41
 0: "K Thermocou
ple":

FILE # 42
NULL FILE
```

The File Lister is most useful with programs containing a label in their first line. Such labels will help the user identify programs. The File Lister enables users to easily update documentation of program cartridges and makes finding a labeled program very easy. Changing variable N in line zero will cause the program to start from a file other than zero.

To use the File Lister:

1. Load file zero from the Software Library.
2. Insert the 9825A cartridge to be listed.
3. Set the 9825A to the desired track by the "trk 0" or the "trk 1" command.
4. Press the 9825A RUN button.

To start at a file other than zero:

1. Load file zero from the Software Library.
2. Insert the 9825A cartridge to be listed.
3. Fetch line zero.
4. Change variable N to one less than the starting file number (e.g., to start at file 6: 5 → N).
5. Store the updated line zero.
6. Set the 9825A to the desired track.
7. Press the 9825A RUN button.

Program Description. This program starts by assigning the value 16 to variable P. Variable P is used to designate the printout device (internal printer). An error linkage is established to label "DoneLister". A buffer and a string variable are also dimensioned. The program then updates the file number variable, N, and finds and identifies that file.

When the file number variable N, is zero, a printout will occur that indicates the current track number. The program will print out file number and the first line of that file until all files have been accessed. Key files, memory files, data files, numeric files, binary files and null files will cause the printout to indicate only file number and the file type.

The key to printing out the first line of each file as line number zero is the use of the list statement to an external device. However, in this case (line 10), the external device is specified by the function "L", line 17. The function "L" defined by line 17 returns the buffer "list" rather than a select code. The line listed can now be examined by examining string variables A$ because the buffer "list" was defined as that string. Use of a non existent format (format seven) deletes extraneous carriage returns and line feeds.

Once the line of program is in a string variable, the program:

1. Finds the line feed.
2. Replaces the line number with line number zero.
3. Prints the string variable A$, 16 characters at a time, on the device with a select code specified by P.

File 0 Program Listing (Identical for both the GPIO and HP-IB Interfaces)

```
0: "Lister":-1→N;16→P;on err "DoneLister";dim A$[106];buf "list",A$,1
1: N+1→N;fdf N;idf A,B,C,D,E,F;if N=0;fmt 2,c,f2.0;wrt P+.2,"Track",E
2: fmt 1,16"-",2/,c,f3.0;wrt P+.1,"FILE #",A
3: if B=6;ldf N,18,10
4: if B=5;wrt P,"KEY FILE";gto 1
5: if B=4;wrt P,"MEMORY FILE";gto 1
6: if B=3;wrt P,"DATA FILE";gto 1
7: if B=2;wrt P,"NUMERIC DATA";gto 1
8: if B=1;wrt P,"BINARY FILE";gto 1
9: if B=0;wrt P,"NULL FILE";gto 1
10: list #'L',18,18
11: red "list",A$
12: "next":len(A$)→Z;" 0"→A$[1,2]
13: for W=1 to 16*int(Z/16) by 16;wrt P,A$[W,W+15];next W
14: buf "list";gto 1;if W<Z;wrt P,A$[W,Z]
15: "DoneLister":if ern=65 and erl=1;dsp "DONE";spc 3;end
16: fmt 3,c,f2.0,c,f2.0;wrt .3,"Error ",char(rom);ern," line",erl;beep;stp
17: "L":ret "list,7"
*6478
```

Auto Loader (File 1)

With the exception of the track specified in line 7, the Auto Loader is identical for both GPIO and HP-IB interfaces. The Auto Loader is designed to search through the user written program and find "cll" statements. Then, the program will load card programs as required by the user program. When the program completes, the user is left with the original program plus all the card programs required from the Software Library. However, the user will have to load any programs not in the Software Library in order to complete the user program.

Detailed instructions on using the Auto Loader are contained in Section II. These instructions will not be repeated here.

Program Description. The key to the Auto Loader is the method of guessing the next available line number. Once this number is found, additional program lines can be loaded from the Library cartridge. The guessing process consists of these steps:

1. Set variable B equal to the number of bytes used to store the last file loaded.
2. Use 10 bytes per line as a starting point to guess the line number.
3. Guess that the next available line is the previous next available line number, 5, plus B/10, i.e., S + B/10 → S).
4. Try to load the next file starting at line number S.
5. If you have guessed wrong and generate an error, go the the error trapping routine and reduce your guess of the next available line by one (S − 1 → S).
6. Go to step 4 and try again until you succeed.

Another important part of the Auto Loader is the portion of the program that searches for "cll" statements. This portion of the program makes use of the list statement to an external device. However, in this case (line 12), the external device is specified by the function, "prog", line 38. The function "prog" defined by line 38 returns the buffer "buf" rather than a select code. The line listed can now be examined by examining string variables A$ because the buffer "buf" was defined as that string.

The program looks at each line to determine if a "cll" statement is on that line. When a "cll" statement is found, the label is isolated. Labels not found in the "directory" represented by string variable C$ will cause the following printout:

```
Not in Directory
cll '------'
```

Label not found

The program uses string variable B$ to keep a running list of Library programs that have been loaded. No Library program will be loaded twice, regardless of how many times it is called or whether both long and short labels are used.

File 1 Program Listings

GPIO (Track 0)

```
 0: "AUTO LOADER":57→S;dim A$[106],B$[50],C$[900],D$[5];sto "Directory"
 1: on err "error";buf "buf",A$,1;"cll '"→D$;"02"→B$
 2: dsp "Insert User Program Tape";stp
 3: ent "Track # ?",T,"Program FILE no. ?",F
 4: trk T;fdf F;idf A,A,B
 5: if A#6;prt "File type ",A;spc 2;jmp -2
 6: S→P;ldf F,S,7
 7: S+B/10→S;dsp "Insert SOFTWARE LIBRARY";stp ;trk 0
 8: fdf 2;idf A,A,B
 9: cfg 1;ldf 2,S,10
10: prnd(S,0)-1→G;cfg 1;S+B/10→S
11: for Z=P to G
12: list #'prog',Z,Z
13: red "buf",A$
14: if (pos(A$,D$)→X)#0;sto 18
15: if X#0;if len(A$)>X;A$[X]→A$;jmp -1
16: next Z
17: sto "end"
18: X+5→X;pos(A$[X],"'")→Y;Y-2→Y
19: if (pos(C$,A$[X,X+Y])→D)=0;prt "Not in Directory",A$[X-5,X+Y+1];sto 15
20: if pos(B$,C$[D-2,D-1])#0;sto 15
21: B$&","&C$[D-2,D-1]→B$
22: val(C$[D-2,D-1])→D
23: "lf":ldf D,S,24
24: fdf D;idf A,A,B;S+B/10→S
25: X+Y→X;sto 15
26: "error":
27: fmt 8,c,2x,c,f2.0
28: on err "error";S-1→S
29: if S<G or ern#31 or rom#0;wrt 16.8,"error",char(rom),ern;spc 2;beep;stp
30: if flg1=0;sto "lf"
31: cfg 1;sto 9
32: "end":spc ;prt "Subprograms from","the Directory","have been loaded"
33: ent "Record or Edit?",A$;if A$#"Edit" and A$#"EDIT" and A$#"edit";sto +3
34: spc ;prt "To remove loader","type in and","execute:"
35: prt "   del 0,56,*";spc 3;end
36: ent "What trk?",T;trk T;ent "What File?",F
37: rcf F,1*57;dsp "Done";end
38: "prog":ret "buf.1"
39: "Directory":"02Parameter Check02PC"→C$
40: "03oo03Simple Output04Serial Output04SO05Parallel Output05PO"&C$→C$
41: C$&"06Interrupt Output06IO07Simple Input07ii08Serial Input08SI"→C$
42: C$&"09Parallel Input09PI02Parameter Check02PC"→C$
43: C$&"10Clear Unit10CU"→C$
44: C$&"11Voltage Serial Output11VS12Voltage Parallel Output12VP"→C$
45: C$&"13Current Serial Output13CS14Current Parallel Output14CP"→C$
46: C$&"15Analog Input15AI16Single Ended Scan16SE"→C$
47: C$&"17Double Ended Scan17DE18External DVM Scanner18ES"→C$
48: C$&"19Motor Output19MO20Set Counter20SC21Read Counter21RC"→C$
49: C$&"22Timer Pulse22TP23Relay Output23RO24Time Interval24TI"→C$
50: C$&"25Frequency Measurement25FM26Option 40 Voltage Control26VC"→C$
51: C$&"27Option 40 Current Control27CC28Arm28AM29Serial Poll"→C$
52: C$&"29SP30Disarm30DM31Time Out31TO32Recycle32RE"→C$
53: C$&"33Reference Word33RW34T Thermocouple34TT35J Thermocouple35JT"→C$
54: C$&"36E Thermocouple36ET37R Thermocouple37RT38S Thermocouple38ST"→C$
55: C$&"39Thermocouple Reference39TR40B Thermocouple40BT"→C$
56: C$&"41K Thermocouple41KT"→C$;sto 1
*12704
```

```
0: "AUTO LOADER":57→S;dim A$[106],B$[50],C$[900],D$[5];sto "Directory"
1: on err "error";buf "buf",A$,1;"cll '"→D$;"02"→B$
2: dsp "Insert User Program Tape";stp
3: ent "Track # ?",T,"Program FILE no. ?",F
4: trk T;fdf F;idf A,A,B
5: if A#6;prt "File type ",A;spc 2;jmp -2
6: S→P;ldf F,S,7
7: S+B/10→S;dsp "Insert SOFTWARE LIBRARY";stp ;trk 1
8: fdf 2;idf A,A,B
9: sfg 1;ldf 2,S,10
10: prnd(S,0)-1→G;cfg 1;S+B/10→S
11: for Z=P to G
12: list #'prog',Z,Z
13: red "buf",A$
14: if (pos(A$,D$,X)#0;sto 18
15: if X#0;if len(A$)>X;A$[X]→A$;jmp -1
16: next Z
17: sto "end"
18: X+5→X;pos(A$[X],"'")→Y;Y-2→Y
19: if (pos(C$,A$[X,X+Y])→D)=0;prt "Not in Directory",A$[X-5,X+Y+1];sto 15
20: if pos(B$,C$[D-2,D-1])#0;sto 15
21: B$&","&C$[D-2,D-1]→B$
22: val(C$[D-2,D-1])→D
23: "lf":ldf D,S,24
24: fdf D;idf A,A,B;S+B/10→S
25: X+Y→X;sto 15
26: "error":
27: fmt 8,c,2x,c,f2.0
28: on err "error";S-1→S
29: if S<G or ern#31 or rom#0;wrt 16.8,"error",char(rom),ern;spc 2;beep;stp
30: if flg1=0;sto "lf"
31: cfg 1;sto 9
32: "end":spc ;prt "Subprograms from","the Directory","have been loaded"
33: ent "Record or Edit?",A$;if A$#"Edit" and A$#"EDIT" and A$#"edit";sto +3
34: spc ;prt "To remove loader","type in and","execute:"
35: prt "   del 0,56,*";spc 3;end
36: ent "What trk?",T;trk T;ent "What File?",F
37: rcf F,1*57;dsp "Done";end
38: "prog":ret "buf.1"
39: "Directory":"02Parameter Check02PC"→C$
40: "03oo03Simple Output04Serial Output04S005Parallel Output05PO"&C$→C$
41: C$&"06Interrupt Output06IO07Simple Input07ii08Serial Input08SI"→C$
42: C$&"09Parallel Input09PI02Parameter Check02PC"→C$
43: C$&"10Clear Unit10CU"→C$
44: C$&"11Voltage Serial Output11VS12Voltage Parallel Output12VP"→C$
45: C$&"13Current Serial Output13CS14Current Parallel Output14CP"→C$
46: C$&"15Analog Input15AI16Single Ended Scan16SE"→C$
47: C$&"17Double Ended Scan17DE18External DVM Scanner18ES"→C$
48: C$&"19Motor Output19MO20Set Counter20SC21Read Counter21RC"→C$
49: C$&"22Timer Pulse22TP23Relay Output23RO24Time Interval24TI"→C$
50: C$&"25Frequency Measurement25FM26Option 40 Voltage Control26VC"→C$
51: C$&"27Option 40 Current Control27CC28Arm28AM29Serial Poll"→C$
52: C$&"29SP30Disarm30DM31Time Out31TO32Recycle32RE"→C$
53: C$&"33Reference Word33RW34T Thermocouple34TT35J Thermocouple35JT"→C$
54: C$&"36E Thermocouple36ET37R Thermocouple37RT38S Thermocouple38ST"→C$
55: C$&"39Thermocouple Reference39TR40B Thermocouple40BT"→C$
56: C$&"41K Thermocouple41KT"→C$;sto 1
*12472
```

CARD SUBPROGRAMS

The remaining pages in this section contain detailed descriptions of the card subprograms recorded on file nos. 2 through 41 of the 14556A cartridge. The format utilized in describing each subprogram is presented below. The individual card subprogram descriptions follow in file number order.

File Number: Call Labels

Abstract of subprogram.

SYNTAX

cll 'XX' (parameter 1, parameter 2, . . . , parameter N)

long or short label

PARAMETERS

Parameter List	Variable	Description	Default Value
parameter 1	p1	Description of parameter	Value assigned by subprogram is parameter not specified by calling program.

PROGRAM EXAMPLES

A typical program line using a "cll" statement to utilize the subprogram.

VARIABLE DESCRIPTION

Description of each variable used in the subprogram.

PROGRAM DESCRIPTION

Description of the function and use of the subprogram.

SPECIAL CONSIDERATIONS

List of items that may effect the use of the subprogram.

USED WITH

List of multiprogrammer plug-in cards that subprogram is designed to call.

DO NOT USE WITH

List of multiprogrammer plug-in cards that subprogram must not call. If any of the listed cards are called, the program will "hang-up."

PROGRAM LISTINGS

Both GPIO and HP-IB program listings are provided. The number of bytes used by each subprogram is given to aid the user in determining the size of the completed program.

Checks if card slot address, card data, and unit number parameters are within limits.

SYNTAX

cll 'Parameter Check' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'PC' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1,p3,etc.	Card Slot Address	None
D_n	p2,p4,etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Parameter Check' (401, A, 407, G, 402, B, 0o)

or

cll 'Parameter Check' (1, 5252, 7, 2525, 2, 7777)

or

cll 'PC' (401, A, 407,7 7, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even number variables are data to be sent to output cards. If these variables are greater than 7777, the program will stop and permit entry of a new value.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

This subprogram is usually entered via a "gsb" from a Library subprogram. However, a "cll" could be made directly to this subprogram. Regardless of how the subprogram is accessed, the program examines parameters p1, p3, etc. to determine if the card slot addresses are correct. New slot addresses can be entered if an out of range number is encountered. To enter the new slot address type the correct slot address and assign it to the variable indicated.

Example: If the display shows:

"Slot address p3 out of range"
Type: 403 → p3 and press EXECUTE
Press CONTINUE

Data variables p2, p4, etc. as well as unit number (pp0) are also tested for limits. Again, if they are out of range, a new value must be assigned before the program will continue.

SPECIAL CONSIDERATIONS

1. Slot addresses must be from 400_{10} to 414_{10} or from zero to 14_{10}.
2. Only data greater than 7777 will be considered as out of range.

USED WITH

all plug-in cards.

PROGRAM LISTINGS

GPIO

```
0: "Parameter Check":
1: "PC":if fls10#1;mdec;dev "MP",9;wti 0,9;sfs 10
2: fmt 9,c,f2.0,/,c,f2.0,2/
3: if int(p0/2)=p0/2;1+p0→p0;0→pp0
4: f/d 0;moct;1→p39
5: if pp39>=400 and pp39<=414;pp39-400→pp39
6: if pp39>=0 and pp39<=14;jmp 2
7: dsp "Slot Address p",p39,"out of range";stp ;jmp -2
8: if p(p39+1)<=7777;jmp 2
9: dsp "Data; p",p39+1,"out of range";stp ;jmp -1
10: if (p39+2→p39)<=p0-1;gto -5
11: if pp0<=15;jmp 2
12: dsp "Unit no.",pp0,"out of range";stp ;jmp -1
13: ret
*21981
```

Bytes used: 466

HP-IB

```
0: "Parameter Check":
1: "PC":if fls10#1;dev "MP",723,"HP-IB",7;sfs 10
2: fmt 9,c,f2.0,/,c,f2.0,2/
3: if int(p0/2)=p0/2;1+p0→p0;0→pp0
4: 1→p39
5: if pp39>=400 and pp39<=414;pp39-400→pp39
6: if pp39>=0 and pp39<=14;jmp 2
7: dsp "Slot address p",p39," out of range";stp ;jmp -2
8: if p(p39+1)<=7777;jmp 2
9: dsp "Data;p",1+p39,"out of range";stp ;jmp -1
10: if (p39+2→p39)<=p0-1;gto -5
11: if pp0<=15;jmp 2
12: dsp "Unit no.",pp0,"out of range";stp ;jmp -1
13: fmt 1,c,z;fmt 2,c,f4.0,c,z;ret
*8261
```

Bytes used: 486

Sends octal data (o to 7777) to the output card (only one card per call) specified. Does not cause 6940B to wait for flag.

SYNTAX

cll 'Simple Output' (S, D, [U])

or

cll 'oo' (S, D, U)

PARAMETERS

Parameter List	Variable	Description	Default Value
S	p1	Card Slot Address	None
D	p2	Card Data	None
U	pp0	Unit No.	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Simple Output' (401, A, 0)

or

cll 'Simple Output' (1, 5252)

or

cll 'oo' (401, A)

In the first example, the long descriptive label and the complete slot address (401) is used, variable data is sent, and the unit number is included. In the next example, the shortened slot address (1) is used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label. complete slot addresses are used, variable data is sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1 — This variable is used for slot address. The value must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot address must be in decimal.

p2 — This variable is the data to be sent to the output card. If this variable is greater than 7777, the program will stop and permit entry at a new value.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. Note that only the SYE mode bit is turned on and the unit number is in octal (dto pp0). Data p2 is then sent to the slot address specified by p1. Finally, a control word with SYE mode bit on is sent to unit 0 (the 6940B) before a return is made to the main program.

SPECIAL CONSIDERATIONS

1. Data sent to the output card must be in octal.
2. Only one card can be called with this subprogram.
3. Since TME is not on, the subroutine will not wait for an external device to time out.

USED WITH

69332A Open Collector Output, 69380A Breadboard Output

PROGRAM LISTING

GPIO

```
0: "Simple Output":
1: "oo":gsb "Parameter Check"
2: if p0>3;beep;dsp "Wrong Number of Parameters";stp
3: wtb "MP",170040+dtopp0
4: wtb "MP",10000*dtop1+p2
5: wtb "MP",170040;mdec
6: ret
*5861
```

Bytes used: 172

HP-IB

```
0: "Simple Output":
1: "oo":gsb "Parameter Check"
2: if p0>3;beep;dsp "Wrong Number of Parameters";stp
3: wrt "MP.2","O",40+dtopp0,"T"
4: wrt "MP.2",char(64+p1),p2,"T"
5: wrt "MP.1","00040T"
6: ret
*12778
```

Bytes used: 186

Sends octal data (0 to 7777) to the output cards specified. Card outputs are updated sequentially.

SYNTAX

cll 'Serial Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'SO' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4,etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Serial Output' (401, A, 407, G, 402, B, 0)

or

cll 'Serial Output' (1, 5252, 7, 2525, 2, 7777)

or

cll 'SO' (401, A, 407, 77, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data to be sent to output cards. If these variables are greater than 7777, the program will stop and permit entry of a new value.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits DTE, SYE, and TME are programmed on by this control word. Therefore, the program will wait for each output to complete its time-out before the next output can be programmed. The program then loops until all slot addresses (p1, p3, etc.) have received the appropriate data (p2, p4, etc.). Finally, a control word with SYE mode bit is sent to unit 0 (the 6940B) before a return is made to the main program.

SPECIAL CONSIDERATIONS

1. Data sent to the call subprogram must be in octal.
2. With DTE and TME mode bits on, cards such as 69331A Digital Output and 69330A Relay Output will send an external gate and wait for an external flag.

USED WITH

69330A Relay Output, 69331A Digital Output, and 69433A Relay Output/Readback.

DO NOT USE WITH

69332A Open Collector Output, 69380A Breadboard Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Serial Output":
1: "SO":gsb "Parameter Check"
2: wtb "MP",170160+dtopp0
3: 1→p39
4: wtb "MP",10000*dtopp39+p(p39+1)
5: if (p39+2→p39)<=p0-1;gto -1
6: wtb "MP",170040;mdec
7: ret
*12295
```

Bytes used: 180

HP-IB

```
0: "Serial Output":
1: "SO":gsb "Parameter Check"
2: wrt "MP.2","0",160+dtopp0,"T"
3: 1→p39
4: wrt "MP.2",char(64+pp39),p(p39+1),"T"
5: if (p39+2→p39)<=p0-1;gto -1
6: wrt "MP.1","00040T"
7: ret
*6293
```

Bytes used: 194

Sends octal data (0 to 7777) to output cards specified. Card outputs are updated simultaneously.

SYNTAX

cll 'Parallel Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'PO' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1,p3,etc.	Card Slot Address	None
D_n	p2,p4,etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Parallel Output' (401, A, 407, G, 402, B, 0)

or

cll 'Parallel Output' (1, 5252, 7, 2525, 2, 7777)

or

cll 'PO' (401, A, 407, 77, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data to be sent to output cards. If these variables are greater than 7777, the program will stop and permit entry of a new value.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits DTE, SYE, and TME are programmed on by this control word. Therefore, the program will wait for each output to complete its time-out before the next output can be programmed. The program then loops until all slot addresses (p1, p3, etc.) have received the appropriate data (p2, p4, etc.). Finally, a control word with SYE mode bit is sent to unit 0 (the 6940B) before a return is made to the main program.

SPECIAL CONSIDERATIONS

1. Data sent to the call subprogram must be in octal.
2. With DTE and TME mode bits on, cards such as 69331A Digital Output and 69330A Relay Output will send an external gate and wait for an external flag.

USED WITH

69330A Relay Output, 69331A Digital Output, and 69433A Relay Output/Readback.

DO NOT USE WITH

69332A Open Collector Output, 69380A Breadboard Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Parallel Output":
1: "PO":gsb "Parameter Check"
2: wtb "MP",170040+dtopp0
3: 1→p39
4: wtb "MP",10000*dtopp39+p(p39+1)
5: if (p39+2→p39)<=p0-1;gto -1
6: wtb "MP",170160,170040;rdec
7: ret
*6433
```

Bytes used: 188

HP-IB

```
0: "Parallel Output":
1: "PO":gsb "Parameter Check"
2: wrt "MP.2","O",40+dtopp0,"T"
3: 1→p39
4: wrt "MP.2",char(64+pp39),p(p39+1),"T"
5: if (p39+2→p39)<=p0-1;gto -1
6: wrt "MP.1","00160T00040T"
7: ret
*3202
```

Bytes used: 202

Sends octal data (0 to 7777) to the output cards specified. Uses the 9825A's interrupt capability to indicate when data transfers with external devices are completed

SYNTAX

cll 'Interrupt Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'IO' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1,p3,etc.	Card Slot Address	None
D_n	p2,p4,etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Interrupt Output' (401, A, 407, G, 402, B, 0)

or

cll 'Interrupt Output' (1, 5252, 7, 2525, 2, 7777)

or

cll 'IO' (401, A, 407, 77, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data to be sent to output cards. If these variables are greater than 7777, the program will stop and permit entry of a new value.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0 with only the SYE mode bit on. The subprogram then loops until all slot addresses (p1, p3, etc.) receive the appropriate data (p2, p4, etc.). Finally, a control word with DTE, SYE, and TME mode bits on is sent. This will permit the cards that were addressed and are timing out to keep the flag to the 9825A Desktop Computer busy until they have all timed out. Before returning to the main program, an on interrupt statement (oni) is executed which establishes a linkage between select code 9 and service routine "Done". An enable interrupt statement (eir) is also executed. When the last card has timed out, a trailing edge of flag will be sent to the 9825A, causing a branch to the specified interupt service routine ("Done").

Following is an example of an interrupt service routine:

```
"Done":

prt "All cards have", "timed out."
moct; wtb 9, 170040; mdec
iret
```

SPECIAL CONSIDERATIONS

1. Data sent to the call subprogram must be in octal.
2. The label "Done" must be used only for the interrupt service routine.

USED WITH

69330A Relay Output, 69331A Digital Output.

DO NOT USE WITH

69332A Open Collector Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Interrupt Output":
1: "IO":gsb "Parameter Check"
2: wtb "MP",170040+dtopp0
3: 1→p39
4: wtb "MP",10000*dtopp39+p(p39+1)
5: if (p39+2→p39)<=p0-1;gto -1
6: wtb "MP",170160;oni "MP","Done";eir "MP";mdec
7: ret
8: "Done":beep;dsp "Done";wait 1000;-1→0;iret
*13917
```

Bytes used: 238

File 6: Interrupt Output (IO)

HP-IB

```
0: "Interrupt Output":
1: "IO":gsb "Parameter Check"
2: rds("MP")→p(p0+3)
3: wrt "MP.2","O",40+dtopp0,"T"
4: 1→p39
5: wrt "MP.2",char(64+pp39),p(p39+1),"T"
6: if (p39+2>p39)<=p0-1;gto -1
7: wrt "MP.1","OOIb0T";uni "HP-IB","Done";eir "HP-IB"
8: ret
9: "Done":beep;dsp "Done";wait 1000;-1→Q;iret
*9065
```

Bytes used: 278

Reads octal data from the input card (only one card per call) specified. The data is returned in octal. Does not cause the 6940B to wait for flag.

SYNTAX

cll 'Simple Input' (S, D [U])

or

cll 'ii' (S, D, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S	p1	Card Slot Address	None
D	p2	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Simple Input' (401, A, 0)

or

cll 'Simple Input' (1, A)

or

cll 'ii' (401, A)

In the first example, the long descriptive label and the complete slot address (401) is used, the data storage location is specified, and the unit number is included. In the next example, the shortened slot address (1) is used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1 — This variable is used for slot the address. The value must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot address must be in decimal.

p2 — This variable is used to store data from the input cards. Data will be returned in octal.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits ISL and SYE are programmed on by this control word. An address word is sent without a gate and the data is read without a gate. Before returning to the main program, a control word with SYE mode bit on is programmed.

SPECIAL CONSIDERATIONS

1. Only one card will be read with this call subprogram, regardless of how many are specified in the parameter list.

USED WITH

69430A Isolated Digital Input and 69480A Breadboard Input.

PROGRAM LISTINGS

GPIO

```
0: "Simple Input":
1: "ii":gsb "Parameter Check"
2: wtb "MP",170240+dtopp0
3: wti 4,10000+dtop1;rdi 4+p2
4: wtb "MP",170040;mdec
5: ret
*5700
```

Bytes used: 130

HP-IB

```
0: "Simple Input":
1: "ii":gsb "Parameter Check"
2: wrt "MP.2","O",240+dtopp0,"T"
3: wrt "MP.2",char(64+p1),"X";red "MP",p2
4: wrt "MP.1","00040T"
5: ret
*18363
```

Bytes used: 146

Returns octal data (IRQ bit removed) from the input cards specified. Cards are read sequentially.

SYNTAX

cll 'Serial Input' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'SI' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Serial Input' (401, A, 407, G, 402, B, 0)

or

cll 'Serial Input' (1, A, 7, G, 2, B,)

or

cll 'SI' (401, A, 407, G, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, the data storage locations are specified, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even numbered variables are used to store data from input cards. The IRQ bit is removed in the call subprogram. Data will be returned in octal.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits ISL, SYE, and TME are programmed on by this control word. Therefore, the program will wait for each input to complete its time-out before the next input can be read. The subprogram then loops until all slot addresses (p1, p3, etc.) have been addressed and read with a gate. Finally, a control word with SYE mode bit is sent to unit 0 (the 6940B) before a return is made to the main program.

SPECIAL CONSIDERATIONS

1. Data returned will be in octal..
2. With TME mode bit on, the 69431A will keep the 6940B flag busy until an external flag returns a trailing edge.

USED WITH

69431A Digital Input.

DO NOT USE WITH

69430A Isolated Digital Input, 69435A Pulse Counter, 69480A Breadboard Input and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Serial Input":
1: "SI":gsb "Parameter Check"
2: wtb "MP",170260+dtopp0
3: 1→p39
4: wti 4,10000+dtopp39;rdb("MP")→p(p39+1)
5: if p(p39+1)>7777;p(p39+1)-100000→p(p39+1)
6: if (p39+2→p39)<=p0-1;gto -2
7: wtb "MP",170040;mdec
8: ret
+15248
```

Bytes used: 232

HP-IB

```
0: "Serial Input":
1: "SI":gsb "Parameter Check"
2: wrt "MP.2","0",260+dtopp0,"T"
3: 1→p39
4: wrt "MP.2",char(64+pp39),"T",char(64+pp39),"X";red "MP",p(p39+1)
5: if p(p39+1)>7777;p(p39+1)-10000→p(p39+1)
6: if (p39+2→p39)<=p0-1;gto -2
7: wrt "MP.1","00040T"
8: ret
+20115
```

Bytes used: 266

Returns octal data (IRQ bit removed) from the input cards specified. The first card that has data ready is read first, followed by the next ready card, etc. The subprogram will not complete until all cards have been read.

SYNTAX

cll 'Parallel Input' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'PI' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Parallel Input' (401, A, 407, G, 402, B, 0)

or

cll 'Parallel Input' (1, A, 7, G, 2, B,)

or

cll 'PI' (401, A, 407, G, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, the data storage locations are specified, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even numbered variables are used to store data from input cards. The IRQ bit is removed in the call subprogram. Data will be returned in octal.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The ISL, SYE mode bits are programmed on by this control word. The the program then loops until all slot addresses (p1, p3, etc.) specified by the call subprogram have been addressed with a gate. Variable p (3 + p0) has a bit set for each card addressed.

The interrupt mode in the 6940B is then enabled by sending a control word with IEN, SYE and TME mode bits on. Then the interrupt linkage is established between interrupts and the program starting at label "IRQ". The subprogram now waits until all cards have interrupted at least once. Each time an interrupt occurs, the program leaves the line labeled "waiting" and branches to the line labeled "IRQ".

Once at "IRQ" a control word is sent with ISL and SYE mode bits on to the unit specified in the call. Each card is now read without gate to determine if it has interrupted (that is, if IRQ is true). If a card has interrupted, a bit is set in variable p (+ p0) and the card is recycled by addressing it with a gate and disarmed. The IRQ bit is then removed from the data.

When all cards have interrupted once, variable p (+ p0) equals variable p (+ p0). The subprogram then programs a control word with SYE mode bit on before returning to the main program.

SPECIAL CONSIDERATIONS

1. Only the first interrupt from a card will cause data to be stored in the specified variable.
2. Any 69431A, 69434A, 69436A, or 69600B card in the system with jumper W6 in (as received from the factory) will be armed and capable of generating an interrupt after the control word with IEN, SYE and TME mode bits on is programmed.
3. Data will be returned in octal with the IRQ bit removed.
4. Jumper W6 does not have to be removed for this call.

USED WITH

69431A Digital Input.

DO NOT USE WITH

69434A Event Sense, 69436A Process Interrput, 69600B Programmable Timer.

PROGRAM LISTINGS

GPIO

```
0: "Parallel Input":
1: "PI":gsb "Parameter Check"
2: "arm":wtb "MP",170240+dtopp0
3: 1→p39
4: 2↑pp39+p(3+p0)→p(3+p0)
5: wtb "MP",10000*dtopp39
6: if (p39+2→p39)<=p0-1;gto -2
7: wtb "MP",170460;oni "MP","IRQ"
8: "waiting":if p(3+p0)#p(2+p0);eir "MP";gto +0
9: "ret":wti 5,40;wtb "MP",170040;mdec;ret
10: "IRQ":wtb "MP",170240+dtopp0
11: 1→p39
12: mdec;if bit(pp39,p(2+p0))=1;gto "increment"
13: moct;wti 4,10000*dtopp39;rdi 4→p(p39+1)
14: if p(p39+1)<10000;gto "increment"
15: mdec;p(p39+1)-100000→p(p39+1);ior(2↑pp39,p(2+p0))→p(2+p0);moct
16: "disarm":wtb "MP",170040+dtopp0,10000*dtopp39
17: "increment":if (p39+2→p39)<=p0-1;gto -5
18: moct;wtb "MP",170460;iret
*18480
```

Bytes used: 644

HP-IB

```
0: "Parallel Input":
1: "PI":gsb "Parameter Check"
2: "arm":wrt "MP.2","0",240+dtopp0,"T"
3: 1→p39
4: 2↑pp39+p(3+p0)→p(3+p0)
5: wrt "MP.2",char(64+pp39),"T"
6: if (p39+2→p39)<=p0-1;gto -2
7: rds("MP")→p(p0+5);wrt "MP.1","00460T";oni "HP-IB","IRQ"
8: "waiting":if p(3+p0)#p(2+p0);eir "HP-IB";gto +0
9: "ret":wrt "MP.1","X00040T";iret
10: "IRQ":rds("MP")→p(p0+5);wrt "MP.2","0",240+dtopp0,"T"
11: 1→p39
12: if bit(pp39,p(2+p0))=1;gto "increment"
13: wrt "MP.2",char(64+pp39),"X";red "MP",p(p39+1)
14: if p(p39+1)<10000;gto "increment"
15: p(p39+1)-10000→p(p39+1);ior(2↑pp39,p(2+p0))→p(2+p0)
16: "disarm":wrt "MP.2","0",40+dtopp0,"T";wrt "MP.1",char(64+pp39),"T"
17: "increment":if (p39+2→p39)<=p0-1;gto -5
18: wrt "MP.1","00460T";iret
*14637
```

Bytes used: 714

File 10: Clear Unit (CU)

Sends logic zeroes to clear all cards in unit (only one unit per call) specified. For example, logic zeroes will clear a Relay Output Card (open all 12 relays).

SYNTAX

cll 'Clear Unit' (U)

or

cll 'CU' (U)

PARAMETERS

Parameter List	Variable	Description	Default Value
p1		Unit Number Address	None

PROGRAM EXAMPLES

cll 'Clear Unit' (4)

or

cll 'Clear Unit' (P)

or

cll 'CU' (P)

The first example uses the long descriptive label to call "Clear Unit". The unit number is given as the constant 4, for the fourth 6941B. The second and third examples use variable P to designate which unit will be cleared. A short label is used in the third example.

VARIABLE DESCRIPTION

p1 Unit number for unit to be cleared (zero data sent to all cards).

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Paramater Check', this subprogram sends a control word to the unit designated by pp0. In this case, pp0 will equal p1, since only one parameter was passed. Next the subprogram sends zero data to all card slots in the unit specified. The subprogram does not wait for any time outs and sends two control words before returning to the main program. The first control word has only the DTE mode bit programmed on. This control word is used to force the 69321B Voltage D/A and 69370A Current D/A to update their second rank of storage to zero. The second control word has only the SYE mode bit on.

SPECIAL CONSIDERATIONS

1. Only one unit can be cleared per call.

USED WITH

All cards.

PROGRAM LISTINGS

GPIO

```
0: "Clear Unit":
1: "CU":gsb "Parameter Check"
2: wti 5,40;wtb "MP",170000+dtop@0
3: 0→p39
4: wtb "MP",10000*dtop39
5: if (p39+1→p39)<=14;gto -1
6: wtb "MP",170100,170040
7: mdec;ret
+5907
```

Bytes used: 174

HP-IB

```
0: "Clear Unit":
1: "CU":gsb "Parameter Check"
2: wrt "MP.2","XO",dtop@0,"T"
3: 0→p39
4: wrt "MP.1",char(64+p39),"T"
5: if (p39+1→p39)<=14;gto -1
6: wrt "MP.1","0100T040T";ret
+32748
```

Bytes used: 172

File 11: Voltage Serial Output (VS)

Sends desired output voltage (−10.240 to +10.235V) in decimal form (e.g., −6.5 for −6.5 volts) to the D/A cards specified. Card outputs are updated sequentially. Note that a 69322A D/A card has 4 output channels which can be individually updated.

SYNTAX

cll 'Voltage Serial Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'VS' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Voltage Serial Output' (401, A, 407, G, 402.3, B, 0)

or

cll 'Voltage Serial Output' (1, −10, 7, 5, 2.3, 4.75)

or

cll 'VS' (401, A, 407, 5, 402.3, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. Note that slot address 402.3 specifies output no. 3 of a 69322A Quad D/A card in slot 2 (see variable description).In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.

All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

Slot addresses for the 69322A Quad D/A Voltage Converter card must have one digit to the right of the decimal point. This digit will specify the output that is to receive new data.

Example:

cll 'Voltage Serial Output (401.1, 10)

indicates Output No. 1 (terminal A)

Slot Address	Output No.	Output Terminal
XXX.1	1	A
XXX.2	2	B
XXX.3	3	D
XXX.4	4	E

Where XXX can be any slot address from 400 to 414.

p2, p4, etc.

All even number variables are data to be sent to output cards. If these variables are greater than 7777, the program will stop and permit entry at a new value.

Data variables are to be expressed in volts. Maximum limits are:

	Negative Output	Positive Output
69321B	−10.240V	+10.235V
69322A	−10.240V	+10.220V

pp0

If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits DTE, SYE, and TME are programmed on by this control word. The program then addresses each card specified, sending it data equivalent to the desired output voltage. Since the DTE mode bit is on, the output is updated immediately. After each card times-out, the next card is sent the appropriate data. Quad D/A Voltage Converters, 69322A, are identified by the non-zero fraction in the slot address variable. A branch to "QUAD" occurs when a 69322A address is sent. This part of the program combines the output data and output number information into one variable (p(p39+41)) for output. After the last card specified has timed-out, the program sends a control word with SYE on to unit 0, the 6940B, before returning to the main program.

SPECIAL CONSIDERATIONS

1. Cards such as 69331A, 69330A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subroutine.
2. Slot addresses that are the result of calculations may have a non-zero fraction that would make them appear to be for 69322A Quad D/A Voltage converters. To prevent this, use the power of ten rounding function (prnd) to remove the fraction (e.g., prnd (A, 0).
3. Current D/A's (69370A's) as well as Voltage D/A's (69321B's) that were previously addressed when DTE was off will update their second rank of storage and their outputs when DTE is programmed on in this call subprogram.

USED WITH

69321B D/A Voltage Converter, 69322A Quad D/A Voltage Converter.

DO NOT USE WITH

69332A Open Collector Output, 69380A Breadboard Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Voltage Serial Output":
1: "VS":esb "Parameter Check"
2: wtb "MP";170160+dtop(p0
3: 1→p39
4: pp39→p(p39+40);if frc(pp39)#0;gto "QUAD"
5: if p(p39+1)>10.235 or p(p39+1)<-10.24;beep;fxd 0;gto +10
6: p(p39+1)/.005→p(p39+41);if p(p39+41)<0;p(p39+41)+4096→p(p39+41)
7: wtb "MP";10000*dtop(p39+40)+dtop(p39+41)
8: if (p39+2→p39)<=p0-1;gto -4
9: wtb "MP";170040;mdec
10: ret
11: "QUAD":if p(p39+1)>10.22 or p(p39+1)<-10.24;sfg 9;gto +4
12: p(p39+1)/.02→p(p39+41);if p(p39+41)<0;p(p39+41)+1024→p(p39+41)
13: (10*frc(p(p39+40))-1)*1024+p(p39+41)→p(p39+41)
14: prnd(pp39,0)→p(p39+40);jmp -7
15: dsp "Data: p";p39+1;"out of range";stp ;gto -10;if fls9;cfg 9;gto -4
*3219
```

Bytes used: 656

HP-IB

```
0: "Voltage Serial Output":
1: "VS":esb "Parameter Check"
2: wrt "MP.2","0",160+dtop(p0,"T"
3: 1→p39
4: pp39→p(p39+40);if frc(pp39)#0;gto "QUAD"
5: if p(p39+1)>10.235 or p(p39+1)<-10.24;beep;fxd 0;gto +10
6: p(p39+1)/.005→p(p39+41);if p(p39+41)<0;p(p39+41)+4096→p(p39+41)
7: wrt "MP.2",char(64+p(p39+40)),dtop(p39+41),"T"
8: if (p39+2→p39)<=p0-1;gto -4
9: wrt "MP.1","00040T"
10: ret
11: "QUAD":if p(p39+1)>10.22 or p(p39+1)<-10.24;sfg 9;gto +4
12: p(p39+1)/.02→p(p39+41);if p(p39+41)<0;p(p39+41)+1024→p(p39+41)
13: (10*frc(p(p39+40))-1)*1024+p(p39+41)→p(p39+41)
14: prnd(p(p39+40),0)→p(p39+40);jmp -7
15: dsp "Data: p";p39+1;"out of range";stp ;gto -10;if fls9;cfg 9;gto -4
*22743
```

Bytes used: 674

File 12: Voltage Parallel Output (VP)

Sends desired output voltage (−10.240V to +10.235V) in decimal form (e.g., 7.5 for +7.5 volts) to the cards specified. 69322A Quad D/A card outputs are updated when the applicable outputs are addressed (e.g., to address channel 2 of 69322A in slot 3 use address 3.2). The 69321B cards are simultaneously updated to their new output voltages.

SYNTAX

cll 'Voltage Parallel Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'VP' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Voltage Parallel Output' (401, A, 407, G, 402.3, B, 0)

or

cll 'Voltage Parallel Output' (1, −10, 7, 5, 2.3, 4.75)

or

cll 'VP' (401, A, 407, 5, 402.3, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. Note that slot address 402.3 specifies output No. 3 of a 69322A Quad D/A card in slot 2 (see variable description). In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.

All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

Slot addresses for the 69322A Quad D/A Voltage Converter card must have one digit to the right of the decimal point. This digit will specify the output that is to receive new data.

Example:

```
cll 'Voltage Serial Output' (401.1, 10)
                             ↖
                             Indicates Output No. 1 (terminal A)
```

Slot Address	Output No.	Output Terminal
XXX.1	1	A
XXX.2	2	B
XXX.3	3	D
XXX.4	4	E

Where XXX can be any slot address from 400 to 414.

p2, p4, etc.

All even number variables are data to be sent to output cards. If these variables are greater than 7777, the program will stop and permit entry at a new value.

Data variables are to be expressed in volts. Maximum limits are:

	Negative Output	Positive Output
69321B	−10.240V	+10.235V
69322A	−10.240V	+10.220V

pp0

If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bit SYE is programmed on by this control word. The program will address each card specified, sending it the data equivalent to the desired output voltage. Since the DTE mode bit is off, the outputs of only the 69321B cards are not updated immediately. Outputs of the 69322A cards, however, are updated immediately. After all cards have received data, another control word is sent with DTE, SYE, and TME mode bits on. The TME mode bit forces the program to wait for all cards to time out before sending the final control word with SYE mode bit on and returning to the main program. The DTE mode bit causes all 69321B outputs to update simultaneously.

Quad D/A Voltage Converters, 69322A, are identified by the non-zero fraction in the slot address variable. A branch to "QUAD1" occurs when a 69322A address is sent. This part of the program combines the output data and output number information into one variable p(p39+41) for output.

SPECIAL CONSIDERATIONS

1. Cards such as 69331A, 69330A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subroutine.
2. Slot addresses that are the result of calculations may have a non-zero fraction that would make them appear to be for 69322A Quad D/A Voltage converters. To prevent this, use the power of ten rounding function (prnd) to remove the fraction (e.g., prnd (A, 0).
3. Current D/A's (69370A's) as well as Voltage D/A's (69321B's) that were previously addressed when DTE was off will update their second rank of storage and their outputs when DTE is programmed on in this call subprogram.

USED WITH

69321A D/A Voltage Converter, 69322A Quad D/A Voltage Converter.

DO NOT USE WITH

69332A Open Collector Output, 69380A Breadboard Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

```
0: "Voltage Parallel Output":
1: "VP":ssb "Parameter Check"
2: wtb "MP",170040+dtop*p0
3: 1→p39
4: pp39→p(p39+40);if frc(pp39)#0;sto "QUAD1"
5: if p(p39+1)>10.235 or p(p39+1)<-10.24;beep;fxd 0;sto +10
6: p(p39+1)/.005→p(p39+41);if p(p39+41)<0;p(p39+41)+4096→p(p39+41)
7: wtb "MP",10000+dtop(p39+40)+dtop(p39+41)
8: if (p39+2→p39)<=p0 1;sto -4
9: wtb "MP",170160,170040;mdec
10: ret
11: "QUAD1":if p(p39+1)>10.22 or p(p39+1)<-10.24;stp 9;sto +4
12: p(p39+1)/.02→p(p39+41);if p(p39+41)<0;p(p39+41)+1024→p(p39+41)
13: (10*frc(p(p39+40))-1)*1024+p(p39+41)→p(p39+41)
14: prnd(pp39,0)→p(p39+40);jmp -7
15: dsp "Data, p";p39+1;" out of range";stp ;sto -10;if fls9;cfg 9;sto -4
*439
```

Bytes used: 668

HP-IB

```
0: "Voltage Parallel Output":
1: "VP":ssb "Parameter Check"
2: wrt "MP.2","O",40+dtop*p0,"T"
3: 1→p39
4: pp39→p(p39+40);if frc(pp39)#0;sto "QUAD1"
5: if p(p39+1)>10.235 or p(p39+1)<-10.24;beep;fxd 0;sto +10
6: p(p39+1)/.005→p(p39+41);if p(p39+41)<0;p(p39+41)+4096→p(p39+41)
7: wrt "MP.2",chr(64+p(p39+40))+dtop(p39+41),"T"
8: if (p39+2→p39)<=p0-1;sto -4
9: wrt "MP.1","00160T00040T"
10: ret
11: "QUAD1":if p(p39+1)>10.22 or p(p39+1)<-10.24;stp 9;sto +4
12: p(p39+1)/.02→p(p39+41);if p(p39+41)<0;p(p39+41)+1024→p(p39+41)
13: (10*frc(p(p39+40))-1)*1024+p(p39+41)→p(p39+41)
14: prnd(p(p39+40),0)→p(p39+40);jmp -7
15: dsp "Data, p";p39+1;"out of range";stp ;sto -10;if fls9;cfg 9;sto -4
*29019
```

Bytes used: 684

File 13: Current Serial Output (CS)

Sends desired output current (0 to 20.475mA) in decimal form (e.g., 4.5 for +4.5mA) to the cards specified. Cards are updated sequentially to their new output currents.

SYNTAX

cll 'Current Serial Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'CS' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Current Serial Output' (401, A, 407, G, 402, B, 0)

or

cll 'Current Serial Output' (1, 10, 7, 5, 2, 4.75)

or

cll 'CS' (401, A, 407, 5, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data to be sent to output cards. Data variables are in milliamps. If these variables are greater than 20.475 or less than 0, the program will stop and permit entry at a new value.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits DTE, SYE, and TME are programmed on by this control word. The program then addresses each card specified, sending it data equivalent to the desired output current. Since the DTE mode bit is on, the output is updated immediately. After each card times-out, the next card is sent the appropriate data. After the last card specified has timed-out, the program sends a control word with SYE on to unit 0, the 6940B, before returning to the main program.

SPECIAL CONSIDERATIONS

1. Cards such as 69331A, 69330A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subroutine.
2. Negative current values may cause data to be sent to cards not specified in the call sequence.
3. Voltage D/A's (69321B's) as well as Current D/A's (69370A's) that were previously addressed when DTE was off will update their second rank of storage and their output when DTE programmed on in this call subprogram.

USED WITH

69370A D/A Current Converter.

DO NOT USE WITH

69332A Open Collector Output, 69380A Breadboard Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Current Serial Output":
1: "CS":gsb "Parameter Check"
2: wtb "MP",170160+dtopp0
3: 1→p39
4: if p(p39+1)<=20.475 and p(p39+1)>=0;gto +2
5: beep;fxd 0;dsp "Data p",p39+1,"out of range";stp ;gto -1
6: wtb "MP",10000*dtopp39+dto(p(p39+1)/.005)
7: if (p39+2→p39)<=p0-1;gto -3
8: wtb "MP",170040;mdec
9: ret
*30318
```

Bytes used: 292

File 13: Current Serial Output (CS)

HP-IB

```
0: "Current Serial Output":
1: "CS":ssb "Parameter Check"
2: wrt "MP.2","U",160+dtupp0,"I"
3: 1→p39
4: if p(p39+1)<=20.475 and p(p39+1)>=0;gto +2
5: beep;if p0 0;dsp "Data p",p39+1,"out of range";stp ;gto -1
6: wrt "MP.2",char(64+pp39),dto(p(p39+1)/.005),"T"
7: if (p39+2→p39)<=p0-1;gto -3
8: wrt "MP.1","U0040T"
9: ret
*R576
```

Bytes used: 306

File 14: Current Parallel Output (CP)

Sends desired output current (0 to 20.475mA) in decimal form (e.g., 4.5 for +4.5mA) to the cards specified. The cards are simultaneously updated to their new output currents.

SYNTAX

cll 'Current Parallel Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'CP' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Current Parallel Output' (401, A, 407, G, 402, B, 0)

or

cll 'Current Parallel Output' (1, 10, 7, 5, 2, 4.7)

or

cll 'CP' (401, A, 407, 5, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used forthe "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data to be sent to output cards. Data variables are in milliamps. If these variables are greater than 20.475 or less than 0, the program will stop and permit entry at a new value.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits DTE, and SYE are programmed on by this control word. The program then addresses each card specified, sending it the data equivalent to the desired output current. Since the DTE mode bit is on, the output is updated immediately. After all cards have received data, another control word is sent with DTE, SYE and TME mode bits on. The TME mode bits forces the program to wait for all cards to time out before sending the final control word with SYE mode bit on and returning to the main program.

SPECIAL CONSIDERATIONS

1. Cards such as 69331A, 69330A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subroutine.
2. Slot addresses that are the result of calculations may have a non-zero fraction that would make them appear to be for 69322A Quad D/A Voltage converters. To prevent this, use the power of ten rounding function (prnd) to remove the fraction (e.g., prnd (A, 0).
3. Voltage D/A's (69321B's) as well as Current D/A's (69370A's) that were previously addressed when DTE was off will update their second rank of storage and their outputs when DTE is programmed on in this call subprogram.

USED WITH

69370A D/A Current Converter.

DO NOT USE WITH

69332A Open Collector Output, 69380A Breadboard Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Current Parallel Output":
1: "CP":gsb "Parameter Check"
2: wtb "MP",170040+dtopp0
3: 1→p39
4: if p(p39+1)<=20.475 and p(p39+1)>=0;gto +2
5: beep;fxd 0;dsp "Data p",p39+1,"out of range";stp ;gto -1
6: wtb "MP",10000+dtopp39+dto(p(p39+1)/.005)
7: if (p39+2→p39)<=p0-1;gto -3
8: wtb "MP",170160,170040;mdec
9: ret
*1e696
```

Bytes used: 300

File 14: Current Parallel Output (CP)

HP-IB

```
0: "Current Parallel Output":
1: "CP":gsb "Parameter Check"
2: wrt "MP.2","O",40+dtopp0,"T"
3: 1→p39
4: if p(p39+1)<=20.475 and p(p39+1)>=0;gto +2
5: beep;fxd 0;dsp "Data p",p39+1,"out of range";stp ;gto -1
6: wrt "MP.2",char(64+pp39),dto(p(p39+1)/.005),"T"
7: if (p39+2→p39)<=p0-1;gto -3
8: wrt "MP.1","00160T00040T"
9: ret
*20323
```

Bytes used: 314

Reads voltage data (−10 to +10V) from the cards specified. Returned data is in decimal form (e.g., −5.5 for −5.5 volts). Cards are read sequentially.

SYNTAX

cll 'Analog Input' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'AI' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Analog Input' (401, A, 407, G, 402, B, 0)

or

cll 'Analog Input' (1, A, 7, G, 2, B,)

or

cll 'AI' (401, A, 407, G, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, the data storage locations are specified, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even number variables are used to store data from input cards. Data will be returned in decimal and will indicate the voltage read.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits ISL, SYE, and TME are programmed on by this control word. The program then reads each card speficied, one at a time, storing the data (after conversion to volts) in the variable specified. A control word is sent, after the last card is read, that has SYE mode bit on, to unit 0. The call subroutine then returns to the main program.

SPECIAL CONSIDERATIONS

1. The call subroutine assumes the 69421A is a standard ±10 volt card. To modify the program for other options, line 7 must be changed as follows:

(for option 102) $0.05p(p39+1) \rightarrow p(p39+1)$
(for option 001) $0.0005p(p39+1) \rightarrow p(p39+1)$

USED WITH

69421A Voltage Monitor, 69422A High Speed A/D.

DO NOT USE WITH

69423A Low Level A/D and Scanner.

PROGRAM LISTINGS

GPIO

```
0: "Analog Input":
1: "AI":gsb "Parameter Check"
2: wtb "MP",170260+dtopp0
3: 1→p39
4: wti 4,10000+dtopp39;rdb("MP")→p(p39+1)
5: if p(p39+1)>7777;p(p39+1)-100000→p(p39+1)
6: otdp(p39+1)→p(p39+1);if p(p39+1)>2047;p(p39+1)-4096→p(p39+1)
7: .005p(p39+1)→p(p39+1)
8: if (p39+2→p39)(=p0-1;gto -4
9: wtb "MP",170040;mdec
10: ret
+21145
```

Bytes used: 334

HP-IB

```
0: "Analog Input":
1: "AI":gsb "Parameter Check"
2: wrt "MP.2","O",260+dtopp0,"T"
3: 1→p39
4: wrt "MP.2",char(64+pp39),"T",char(64+pp39),"X";red "MP",p(p39+1)
5: if p(p39+1)>7777;p(p39+1)-10000→p(p39+1)
6: otdp(p39+1)→p(p39+1);if p(p39+1)>2047;p(p39+1)-4096→p(p39+1)
7: .005p(p39+1)→p(p39+1)
8: if (p39+2→p39)(=p0-1;gto -4
9: wrt "MP.1","00040T"
10: ret
+6040
```

Bytes used: 368

File 16: Single Ended Scan (SE)

Scans and read voltages from multiplexed dc sources. Measured voltages in decimal form are read from 69421A or 69422A cards. The 69330A or 69433A cards are used to scan the dc voltages to be measured. All readings are stored in r-variables. The voltages scanned must share a single common or ground wire.

SYNTAX

cll 'Single Ended Scan' (Sv, R, C, [U])

or

cll 'SE' (Sv, R, C, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
Sv	p1	Voltage Monitor Slot Address	None
R	p2	No. of Readings	None
C	p3	Start Channel	Unit 0
U	p4	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Single Ended Scan' (403, 98, 14, 0)

or

cll 'Single Ended Scan' (3, 98, 14)

cll 'SE' (403, 98, 14)

In the first example, the long descriptive label and the complete slot addresses (403) is used, the number of readings and the starting channel are specified, and the unit number is included. In the next example, the shortened slot address (3) is used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1	Card slot address for the Voltage Monitor Card, 69421A or 69422A High Speed A/D card. The value must be from 400 to 413 or from 0 (zero) to 13, where 1 is understood to be slot address 401, 2 is 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot address must be in decimal. The first Relay Output card, 69330A or Relay Output/Readback Card, 69433A must be in the next higher slot address. All relay cards used for scanning must be next to one another.
p2	The total number of readings must not exceed 168, since that is the maximum obtainable from one mainframe (14 cards). Minimum value is one reading.
p3	Start channel number can be any value from zero to 168.

Note: Program will adjust maximum limit for start channel plus number of readings to reflect slot address for first relay card.

PROGRAM DESCRIPTION

The subprogram starts with a "cll" to 'Parameter Check'. Parameters sent are the card slot address for the Voltage Monitor, a dummy variable (zero) and the unit address. If either the slot address (p1) or the unit number (p4) are out of limits, the program will stop and wait for correct values to be entered.

Upon returning from 'Parameter Check', the program checks the maximum number of readings registered. If more than the available number of readings are requested, the program stops and waits until the number of readings requested is corrected. Next, the starting channel is checked and if it is not between zero and the available number of channels, the program again stop and waits until the start channel number is corrected. The final check is on the total of start channel plus total number of readings. If this number exceeds the available number of channels, the program again stops and waits until the total is corrected. The available number of channels is determined by assuming there are relay cards in all card slots to the right of the Voltage Monitor card.

All relays are now programmed open to prevent two sources from being shorted together. Scanning is then begun by calculating the first card and first relay to be closed. Since a control word with DTE, SYE and TME mode bits has been programmed, the program will wait for each relay to settle before programming the control word with ISL, SYE, and TME mode bits on. Once this control word is programmed, the Voltage Monitor card is then addressed and read with a gate. The reading is converted to volts and stored in an r-variable, rp39; where p39 represents the channel number. The relay that was closed for the last reading is now opened by sending a control word with DTE, SYE and TME mode bits and sending zero data to the appropriate relay card. The program then repeats the relay closure, voltage measurement, relay open cycle until all channels specified have been read. After the final reading, a control word with SYE mode bit on is sent to unit 0, the 6940B, before returning to the main program.

SPECIAL CONSIDERATIONS

1. All relay cards used for scanning must be mounted in the card slots immediately to the right of the Voltage Monitor card.
2. Cards such as 69330A, 69331A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subprogram.
3. The call subprogram assumes the 69421A is a standard ±10 volt card. To modify the program for other options, line 19 must be changed as follows:

(for option 102) .05rp39 → rp39
(for option 001) .0005rp39 → rp39

4. Voltage D/A's (69321B's) and Current D/A's (69370A's) that were previously addressd when DTE was off will update their second rank of storage and their outputs when DTE is programmed on in this call subprogram.

USED WITH (See Figure 3-1)

A/D Cards: 69421A Voltage Monitor, 69422A High Speed A/D

Scanner Cards: 69330A Relay Output, 69433A Relay Output/Readback

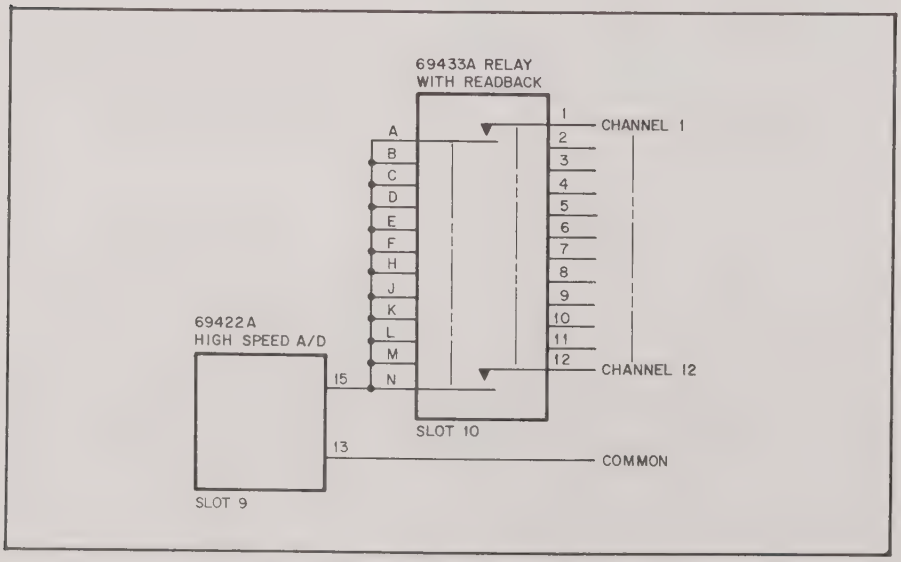


Figure 3-1. Single Ended Scan Connection Diagram

PROGRAM LISTINGS

GPIO

```
0: "Single Ended Scan":
1: "SE";cll 'Parameter Check'(p1,0,p4);12*(p1+1)→p5
2: if p2<1 or p2>168-p5;prt "Total readings:";p2;"out of range";stp ;jmp 0
3: if p3<0 or p3>168-p5;prt "Starting channel:";p3;"out of range";stp ;jmp 0
4: if p2+p3>168-p5;prt "Last channel:";p2+p3;">168";stp ;jmp -2
5: wtb "MP";170140+dtop4
6: prnd(.49+(p2+p3)/12,0)→p6
7: 1→p39
8: wtb "MP";10000*dto(p1+p39)
9: if (p39+1→p39)<=p6;sto -1
10: wtb "MP";170160+dtop4
11: p3→p39
12: prnd(.51+p39-12,0)→p5
13: wtb "MP";10000*dto(p1+p5)+dto(2↑(p39mod12))
14: wtb "MP";170260+dtop4
15: wti 4,10000+dtop1;rdb("MP")→rp39
16: wtb "MP";170160+dtop4,10000*dto(p1+p5)
17: if rp39>7777;rp39-100000→rp39
18: otdrp39→rp39;if rp39>2047;rp39-4096→rp39
19: .005rp39→rp39
20: if (p39+1→p39)<(p2+p3;sto -8
21: wtb "MP";170040;mdec
22: ret
+19031
```

Bytes used: 816

HP-IB

```
0: "Single Ended Scan":
1: "SE";cll 'Parameter Check'(p1,0,p4);12*(p1+1)→p5
2: if p2<1 or p2>168-p5;prt "Total readings:";p2;"out of range";stp ;jmp 0
3: if p3<0 or p3>168-p5;prt "Starting channel:";p3;"out of range";stp ;jmp 0
4: if p2+p3>168-p5;prt "Last channel:";p2+p3;">168";stp ;jmp -2
5: wrt "MP.2","O",140+dtop4,"T"
6: prnd(.49+(p2+p3)/12,0)→p6
7: 1→p39
8: wrt "MP.2",char(64+p1+p39),"T"
9: if (p39+1→p39)<=p6;sto -1
10: wrt "MP.2","O",160+dtop4,"T"
11: p3→p39
12: prnd(.51+p39/12,0)→p5
13: wrt "MP.2",char(64+p1+p5),dto(2↑(p39mod12)),"T"
14: wrt "MP.2","O",260+dtop4,"T"
15: wrt "MP.2",char(64+p1),"TX";red "MP";rp39
16: wrt "MP.2","O",160+dtop4,"T",char(64+p1+p5),"T"
17: if rp39>7777;rp39-10000→rp39
18: otdrp39→rp39;if rp39>2047;rp39-4096→rp39
19: .005rp39→rp39
20: if (p39+1→p39)<(p2+p3;sto -8
21: wrt "MP.1","00040T"
22: ret
+29652 .
```

Bytes used: 872

File 17: Double Ended Scan (DE)

Scans and reads voltages from multiplexed DC sources. Measured voltages in decimal form are read from 69421A or 69422A cards. The 69330A or 69433A cards are used to scan the DC voltages to be measured. The voltages scanned need not share a single common or ground wire. All readings are stored in r-variables.

SYNTAX

cll 'Double Ended Scan' (Sv, R, C,)

or

cll 'DE' (Sv, R, C, U)

PARAMETERS

Parameter List	Variable	Description	Default Value
Sv	p1	Voltage Monitor Slot Address	None
R	p2	No. of Readings	None
C	p3	Start Channel	None
U	p4	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Double Ended Scan' (403, 98, 14, 0)

or

cll 'Double Ended Scan' (3, 98, 14)

or

cll 'DE' (403, 98, 14)

In the first example, the long descriptive label and the complete slot address (403) is used, the number of readings and the starting channel are speficied, and the unit number is included. In the next example, the shortened slot address (3) is used and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1	Card slot address for the Voltage Monitor Card, 69421A or 69422A High Speed A/D card. The value must be from 400 to 413 or from 0 (zero) to 13, where 1 is understood to be slot address 401, 2 is 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot address must be in decimal. The first Relay Output card, 69330A or Relay Output/Readback Card, 69433A must be in the next higher slot address. All relay cards used for scanning must be next to one another.
p2	The total number of readings must not exceed 84, since that is the maximum obtainable from one mainframe (14 cards). Minimum value is one reading.
p3	Start channel number can be any value from zero to 84.

Note: Program will adjust maximum limit for start channel plus number of readings to reflect slot address for first relay card.

PROGRAM DESCRIPTION

The subprogram starts with a "cll" to 'Parameter Check'. Parameters sent are the card slot address for the Voltage Monitor, a dummy variable (zero) and the unit address. If either the slot address (p1) or the unit number (p4) are out of limits, the program will stop and wait for correct values to be entered.

Upon returning from 'Parameter Check', the program checks the maximum number of readings requested. If more than the available number of readings are requested, the program stops and waits until the number of readings requested is corrected. Next, the starting channel is checked and if it is not between zero and the available number of channels, the program again stop and waits until the start channel number is corrected. The final check is on the total of start channel plus total number of readings. If this number exceeds the available number of channels, the program again stops and waits until the total is corrected. The available number of channels is determined by assuming there are relay cards in all card slots to the right of the Voltage Monitor card.

All relays are now programmed open to prevent two sources from being shorted together. Scanning is then begun by calculating the first card and first relay to be closed. Since a control word with DTE, SYE and TME mode bits has been programmed, the program will wait for each relay to settle before programming the control word with ISL, SYE, and TME mode bits on. Once this control word is programmed, the Voltage Monitor card is then addressed and read with a gate. The reading is converted to volts and stored in an r-variable, rp39; where p39 represents the channel number. The relay that was closed for the last reading is now opened by sending a control word with DTE, SYE and TME mode bits and sending zero data to the appropriate relay card. The program then repeats the relay closure, voltage measurement, relay open cycle until all channels specified have been read. After the final reading, a control word with SYE mode bit on is sent to unit 0, the 6940B, before returning to the main program.

SPECIAL CONSIDERATIONS

1. All relay cards used for scanning must be mounted in the card slots immediately to the right of the Voltage Monitor card.
2. Cards such as 69330A, 69331A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subprogram.
3. The call subprogram assumes the 69421A is a standard ±10 volt card. To modify the program for other options, line 20 must be changed as follows:

(for option 102) .05rp39 → rp39
(for option 001) .0005rp39 → rp39

4. Voltage D/A's (69321B's) and Current D/A's (69370A's) that were previously addressd when DTE was off will update their second rank of storage and their outputs when DTE is programmed on in this call subprogram.

USED WITH (See Figure 3-2)

A/D Cards: 69421A Voltage Monitor, 69422A High Speed A/D

Scanner Cards: 69330A Relay Output, 69433A Relay Output/Readback

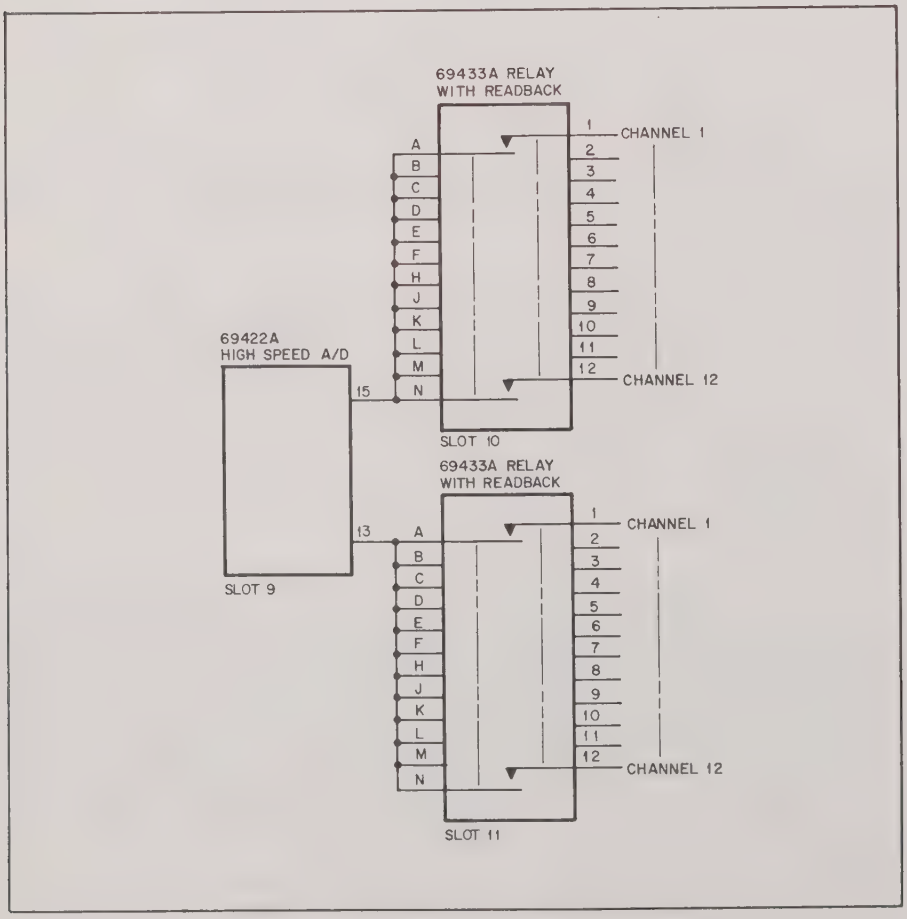


Figure 3-2. Double Scan Connection Diagram

PROGRAM LISTINGS

GPIO

```
0: "Double Ended Scan":
1: "DE":cll 'Parameter Check'(p1;0;p4);12*int((14-p1)/12)→p5
2: if p2<1 or p2>84-p5;prt "Total readings:",p2,"out of range";stp ;jmp 0
3: if p3<0 or p3>84-p5;prt "Starting channel:",p3,"out of range";stp ;jmp 0
4: if p2+p3>84-p5;prt "Last channel:",p2+p3,">84";stp ;jmp -2
5: wtb "MP";170140+dtop4
6: 2*prnd(.49+(p2+p3)/12,0)→p6
7: 2*prnd(.49+p3/12,0)→p7
8: 1→p39
9: wtb "MP";10000*dto(p1+p39)
10: if (p39+1→p39)<=p6;sto -1
11: wtb "MP";170160+dtop4
12: p3→p39
13: 2*prnd(.51+p39/12,0)-1→p5;dto(2↑(p39mod12))→p8
14: wtb "MP";10000*dto(p1+p5)+p8;10000*dto(p1+p5+1)+p8
15: wtb "MP";170260+dtop4
16: wti 4,10000*dtop1;rdb("MP")→rp39
17: wtb "MP";170140+dtop4;10000*dto(p1+p5);10000*dto(p1+p5+1);170160+dtop0
18: if rp39>7777;rp39-100000→rp39
19: otdrp39→rp39;if rp39>2047;rp39-4096→rp39
20: .005rp39→rp39
21: if (p39+1→p39)<p2+p3;sto -8
22: wtb"MP";170040;mdec
23: ret
+36992
```

Bytes used: 934

HP-IB

```
0: "Double Ended Scan":
1: "DE":cll 'Parameter Check'(p1;0;p4);12*int((14-p1)/12)→p5
2: if p2<1 or p2>84-p5;prt "Total readings:",p2,"out of range";stp ;jmp 0
3: if p3<0 or p3>84-p5;prt "Starting channel:",p3,"out of range";stp ;jmp 0
4: if p2+p3>84-p5;prt "Last channel:",p2+p3,">84";stp ;jmp -2
5: wrt "MP.2","O";140+dtop4,"T"
6: 2*prnd(.49+(p2+p3)/12,0)→p6
7: 2*prnd(.49+p3/12,0)→p7
8: 1→p39
9: wrt "MP.2";char(64+p1+p39),"T"
10: if (p39+1→p39)<=p6;sto -1
11: wrt "MP.2","O",160+dtop4,"T"
12: p3→p39
13: 2*prnd(.51+p39/12,0)-1→p5;dto(2↑(p39mod12))→p8
14: wrt "MP.2";char(64+p1+p5),p8,"T";char(64+p1+p5+1),p8,"T"
15: wrt "MP.2","O",260+dtop4,"T"
16: wrt "MP.2";char(64+p1),"TX";red "MP";rp39
17: wrt "MP.2","O",160+dtop4,"T";char(64+p1+p5),"T";char(64+p1+p5+1),"T"
18: if rp39>7777;rp39-10000→rp39
19: otdrp39→rp39;if rp39>2047;rp39-4096→rp39
20: .005rp39→rp39
21: if (p39+1→p39)<p2+p3;sto -8
22: wrt "MP.1","00040T"
23: ret
+7677
```

Bytes used: 984

File 19: Motor Output (MO)

Sends signed decimal data to the card specified. Positive numbers will cause stepping motors to rotate in one direction, and negative numbers in the reverse direction. Data greater than ±2047 steps will cause a stop.

SYNTAX

cll 'Motor Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'MO' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data in Signed Decimal	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Motor Output' (401, A, 407, G, 402, B, 0)

or

cll 'Motor Output' (1, 100, 7, 500, 2, −1000)

or

cll 'MO' (401, A, 407, 500, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data (signed decimal)to be sent to output cards. Data variables are in milliamps. If these variables are more positive than +2047 or more negative than −2047, the program will stop and permit entry at a new value.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram checks the data for sign and magnitude and converts the data to signed octal. A control word is sent to the unit designated by pp0 after pp0 is converted to octal (dto pp0). The converted data is then sent to the appropriate card. After all cards specified in the call have been addressed and given new data, the call subprogram sends a control word with SYE mode bit on to unit zero. This call program does not wait for all cards to time out prior to returning to the main program.

SPECIAL CONSIDERATIONS

1. Other call subprograms that turn on the TME mode bit (e.g., Serial Output) will not complete until the stepper motors programmed with Motor Output have timed out.
2. "Time Out" routine may be used to have system await motor completion.
3. Parameters out of range (e.g., more than 2047 steps programmed) will cause a halt. The program will not continue until the data is within range.

USED WITH

69335A Stepping Motor Control Card.

PROGRAM LISTINGS

GPIO

```
0: "Motor Output":
1: "MO":
2: gsb "Parameter Check"
3: 1→p39
4: if abs(p(p39+1))>2047;fxd 0;dsp "Data;p",p39+1," out of range";stp ;jmp 0
5: if p(p39+1)<0;dtoabs(p(p39+1))+4000→p42;jmp 2
6: if p(p39+1)>=0;dtop(p39+1)→p42
7: wtb "MP",170040+dtopp0
8: wtb "MP",10000*dtopp39+p42;if (p39+2→p39)<=p0-1;gto -4
9: wtb "MP",170040;mdec
10: ret
*28807
```

Bytes used: 330

HP-IB

```
0: "Motor Output":
1: "MO":
2: gsb "Parameter Check"
3: 1→p39
4: if abs(p(p39+1))>2047;fxd 0;dsp "Data;p",p39+1,";out of range";stp ;jmp 0
5: if p(p39+1)<0;dtoabs(p(p39+1))+4000→p42;jmp 2
6: if p(p39+1)>=0;dtop(p39+1)→p42
7: wrt "MP.2","O";40+dtopp0;"T"
8: wrt "MP.2",char(64+pp39),p42,"T";if (p39+2→p39)<=p0-1;gto -4
9: wrt "MP.1","00040T"
10: ret
*20291
```

Bytes used: 344

Presets specified counters to the decimal count sent as data. Maximum count that card can be set to is 4095.

SYNTAX

cll 'Set Counter' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'SC' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data in Decimal	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Set Counter' (401, A, 407, G, 402, B, 0)

or

cll 'Set Counter' (1, 1000, 7, 2000, 2, 3000)

or

cll 'SC' (401, A, 407, 2000, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even number variables are decimal data to be sent to output cards. If these variables are greater than 4095, the program will stop and permit entry at a new value.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The specified counters are set to the value specified by the data parameters. The data is converted from decimal to octal before being sent to the card. After all cards have been set, the program sends a control word with SYE mode bit on to unit zero, the 6940B. The program then returns to the main program.

SPECIAL CONSIDERATIONS

1. Maximum number that the counters can be set to is 4095 (decimal).

USED WITH

69435A Pulse Counter.

PROGRAM LISTINGS

GPIO

```
0: "Set Counter":
1: "SC":gsb "Parameter Check"
2: wtb "MP";170040+dtopp0
3: 1→p39
4: if p(p39+1)>=0 and p(p39+1)<4096;gto +2
5: beep;fxd 0;disp "data p";p39;"out of range";stp ;gto -1
6: wtb "MP";10000+dtopp39+dtop(p39+1)
7: if (p39+2→p39)<=p0-1;gto -3
8: wtb "MP";170040
9: mdec;ret
*18986
```

Bytes used: 270

HP-IB

```
0: "Set Counter":
1: "SC":gsb "Parameter Check"
2: wrt "MP.2","0",40+dtopp0,"T"
3: 1→p39
4: if p(p39+1)>=0 and p(p39+1)<4096;gto +2
5: beep;fxd 0;disp "data p";p39+1;"out of range";stp ;gto -1
6: wrt "MP.2",char(64+pp39),dtop(p39+1),"T"
7: if (p39+2→p39)<=p0-1;gto -3
8: wrt "MP.1","00040T"
9: ret
*23927
```

Bytes used: 290

File 21: Read Counter (RC)

Reads the specified counter cards. Returned data is in decimal form and is stored in the specified variables.

SYNTAX

cll 'Read Counter' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'RC' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Read Counter' (401, A, 407, G, 402, B, 0)

or

cll 'Read Counter' (1, A, 7, G, 2, B)

or

cll 'RC' (401, A, 407, G, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, the data storage locations are specified, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even numbered variables are used to store data from input cards. Data will be returned in decimal.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits ISL and SYE are programmed on by this control word. The cards specified are read. Their data is converted to decimal and stored in the variables specified by the call. After the last data item has been converted and stored, the program sends a control word with SYE mode bit on to unit zero, the 6940B, before returning to the main program.

SPECIAL CONSIDERATIONS

1. The 69435A Counter Card is read "on-the-fly", therefore, there is a finite possibility that the data will be read as the count is being updated.

USED WITH

69435A Pulse Counter Card.

PROGRAM LISTINGS

GPIO

```
0: "Read Counter":
1: "RC":gsb "Parameter Check"
2: wtb "MP",170240+dtopp0
3: 1→p39
4: wti 4,10000*dtopp39;otdrdb("MP")→p(p39+1)
5: if (p39+2→p39)<=p0-1;gto -1
6: wtb "MP",170040;mdec
7: ret
*11761
```

Bytes used: 186

HP-IB

```
0: "Read Counter":
1: "RC":gsb "Parameter Check"
2: wrt "MP.2","O",240+dtopp0,"T"
3: 1→p39
4: wrt "MP.2",char(64+pp39),"X";red "MP",p(p39+1);otdp(p39+1)→p(p39+1)
5: if (p39+2→p39)<=p0-1;gto -1
6: wrt "MP.1","00040T"
7: ret
*31288
```

Bytes used: 224

File 22: Timer Pulse (TP)

Sends the time (in seconds) to the card specified. Time interval resolution is 0.001 seconds.

SYNTAX

cll 'Timer Pulse' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'TP' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data in Seconds	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Timer Pulse' (401, A, 407, G, 402, B, 0)

or

cll 'Timer Pulse' (1, 1.50, 7, 3.75, 2, 4.00)

or

cll 'TP' (401, A, 407, 3.75, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data in seconds, to be sent to output cards. If these variables are greater than 4.095, the program will stop and permit entry at a new value. Resolution is one millisecond.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sets the range parameter, p38, to 0.001. This corresponds to inserting jumper W2-D or connecting pins 11 and 4 on the card edge connector. To change resolution to 100 microseconds, change the range parameter, p38, to 0.0001 and connect W2-E on pins 11 and 3 on the edge connector.

A control word with DTE and SYE mode bits is then sent to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Each card specified is then programmed for the appropriate time interval. After the last card has been programmed and without waiting for any cards to time out, a control word with SYE mode bit on is sent to unit zero, the 6940B, before returning to the main program.

SPECIAL CONSIDERATIONS

1. Maximum time interval is 4.095 seconds.
2. Cards such as 69331A, 69330A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subprogram.
3. Voltage D/A cards (69321B's) and current D/A cards (69370A's) that were previously addressed when DTE was off will update their second rank of storage and their outputs when DTE is programmed on in this call subprogram.
4. The system can be forced to wait for the time interval pulse to complete if jumper W3 is installed on 69600B card and the "Time Out" subprogram is called immediately after the "Timer Pulse" call.

USED WITH

69600B Programmable Timer.

PROGRAM LISTINGS

GPIO

```
0: "Timer Pulse":
1: "TP":gsb "Parameter Check"
2: .001→p38;1→p39
3: if p(p39+1)<=.001 and p(p39+1)<=4.095;sto +2
4: fxd 0;beep;dsp "Data p",p39+1,"out of range";stp ;sto -1
5: wtb "MP";170140+dtopp0
6: wtb "MP";10000*dtopp39+int(dto(p(p39+1)/p38))
7: if (p39+2→p39)<=p0-1;sto -4
8: wtb "MP",170040;mdec;ret
+32236
```

Bytes used: 292

File 22: Timer Pulse (TP)

HP-IB

```
0: "Timer Pulse":
1: "TP":gsb "Parameter Check"
2: .001→p38;1→p39
3: if p(p39+1)>=.001 and p(p39+1)<=4.095;gto +2
4: fxd 0;beep;dsp "Data p",p39+1,"out of range";stp ;gto -1
5: wrt "MP.2","O",140+dtopp0,"T"
6: wrt "MP.2",char(64+pp39),int(dto(p(p39+1)/p38)),"T"
7: if (p39+2→p39)<=p0-1;gto -4
8: wrt "MP.1","040T";ret
*4688
```

Bytes used: 306

File 23: Relay Output (RO)

Changes the state of the desired relay (1 through 12) on the specified card. Data must be in the range of 1 to 12 or −12 to −1. Relays are closed for positive numbers and opened for negative numbers.

SYNTAX

cll 'Relay Output' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'RO' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Relay Output' (401, A, 407, G, 402, B, 0)

or

cll 'Relay Output' (1, 4, 7, −3, 2, 6)

or

cll 'RO' (401, A, 407, −3, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even number variables are data to be sent to output cards. The data specifies which individual relay is to change state. Positive data will cause that relay to close, while negative data will cause that relay to open. Data must be in the range of −12 to −1 or 1 to 12.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). Mode bits ISL and SYE are programmed on by this control word. Each card is read to determine the present status of relays. Then, using the present status and the relay to be changed, the relay card is updated. After all cards have been updated and have timed out, the subprogram sends a control word with SYE mode bit on. A return is then made to the main program.

SPECIAL CONSIDERATIONS

1. Data represents the connector pin associated with a specific relay and not the bit number (e.g., data = 2 will close a relay between connector pins 2 and B).
2. Cards such as 69331A, 69330A that have previously been addressed when DTE was off will send an external gate when DTE is programmed on in this call subprogram.
3. Voltage D/A's (69321B's) and Current D/A cards (69370A's) that were previously addressed when DTE was off will update their second rank of storage and their outputs when DTE is programmed on in this call subprogram.

USED WITH

69433A Relay/Readback.

PROGRAM LISTINGS

GPIO

```
0: "Relay Output":
1: "RO";gsb "Parameter Check"
2: 1→p39
3: if abs(p(p39+1))>0 and abs(p(p39+1))<13;gto +2
4: fxd 0;dsp "Data p",p39+1,"out of range";stp ;gto -1
5: wtb "MP",170240+dtopp0
6: wti 4,10000*dtopp39;rdi 4→p38
7: shf(1,1-abs(p(p39+1)))→p37
8: if p(p39+1)>0;band(7777,ior(p37,p38))→p37;gto +2
9: band(band(p38,cmpp37),7777)→p37
10: wtb "MP",170160+dtopp0,10000*dtopp39+p37
11: if (p39+2→p39)<=p0-1;gto -8
12: wtb "MP",170040;mdec;ret
*10222
```

Bytes used: 430

File 23: Relay Output (RO)

HP-IB

```
0: "Relay Output":
1: "RO":gsb "Parameter Check"
2: 1→p39
3: if abs(p(p39+1))>0 and abs(p(p39+1))<13;gto +2
4: fxd 0;beep;dsp "Data p";p39+1;"out of range";stp ;gto -1
5: wrt "MP.2","O",240+dtopp0,"T";moct
6: wrt "MP.2",char(64+pp39),"X";red 723,p38
7: shf(1,1-abs(p(p39+1)))→p37
8: if p(p39+1)>0;bond(7777,ior(p37,p38))→p37;gto +2
9: band(band(p38,cmpp37),7777)→p37
10: wrt "MP.2","O",160+dtopp0,"T",char(64+pp39),p37,"T"
11: if (p39+2→p39)<=p0-1;gto -8
12: wrt "MP.1","040T";mdec;ret
*365
```

Bytes used: 466

Returns the decimal value of an external pulse. The time base (69601B) is assumed to be 100kHz.

SYNTAX

cll 'Time Interval' (S_0, D_0, [U])

or

cll 'TI' (S_0, D_0, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1	Card Slot Address	None
D_n	p2	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None
	p4	Reference Frequency	100000

PROGRAM EXAMPLES

cll 'Time Interval' (401, A, 0)

or

cll 'Time Interval' (1, A)

or

cll 'TI' (401, A)

In the first example, the long descriptive label and the complete slot address (401)is used, the data storage location is specified, and the unit number is included. In the next example, the shortened slot address (1) is used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1 — Used for slot address. The value must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2 — Used to store data from input card. Data is returned in decimal (seconds).

p4	Must equal the external frequency reference applied to pin E of the 69435A edge connector.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram requires two cards: the 69435A Pulse Counter and 69601B Frequency Reference Cards (see the connection diagram of Figure 3-3). The purpose of this subprogram is to time a pulse applied to the 69435A count-up enable input, pin A of the edge connector. The width of the pulse can vary from microseconds to about one hour (4000 seconds). See the chart on Figure 3-3 for appropriate connections.

The subprogram starts with a go sub (gsb) to 'Parameter Check'. The Parameter Check subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check'. this subprogram sends a control word, with SYE mode bit on, to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram also checks the number of parameters passed. If less than three are passed, the program displays: "No return variable in cll statement", and then stops.

If enough parameters are passed, the call subprogram sends a control word with ISL and SYE mode bits on. Then it addresses and reads the 69435A Pulse Counter card. While waiting for a non-zero reading, the program reads the 69435A and displays:

"Waiting for pulse"

A wait of approximately 50 milliseconds occurs after the first non-zero reading from the 69435A. This wait is to ensure that the pulse being measured has completed. The wait (approximately 50msec) is based on the assumption that the reference frequency from the 69601B is 100kHz. Therefore, p4 is 100000 as indicated by the chart in the Time Interval Connection Diagram. Longer pulses require two changes:

1. Connection of a different reference frequency.
2. Change in the variable p4.

For example:

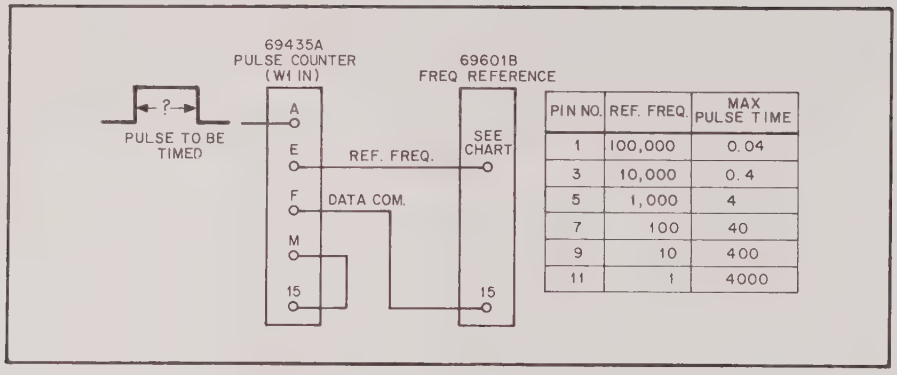
A time interval of approximately 0.6 seconds must be timed. The pulse is connected to pin A of the 69435A Counter Card edge connector. Then, pin E of the 69435A is connected to pin 5 of the 69601B Frequency Reference card edge connector. This connection will give a reference frequency of 1000Hz and therefore a resolution of one millisecond. Line 2 in the subprogram must be changed from 100000 → p4 to 1000 → p4. With this connection, pulses from milliseconds to 4 seconds can be measured.

SPECIAL CONSIDERATIONS

1. Reference Frequncy parameter, p4, is 100000, therefore the reference frequency is assumed to be 100kHz. If a different reference frequency is desired, p4 can be modified to correspond with the proper frequency.
2. Pulse to be measured is TTL compatible.

USED WITH

69435A Pulse Counter, 69601B Frequency Reference.



PIN NO.	REF. FREQ.	MAX PULSE TIME
1	100,000	0.04
3	10,000	0.4
5	1,000	4
7	100	40
9	10	400
11	1	4000

Figure 3-3. Time Interval Connection Diagram

PROGRAM LISTINGS

GPIO

```
0: "Time Interval":
1: "TI":gsb "Parameter Check"
2: 100000→p4
3: if p0<3;dsp "No return variable in cll statement";beep;end
4: wtb "MP";170040+dtopp0;10000*dtop1;170240+dtopp0;10000*dtop1
5: dsp "Waiting for pulse";if rdi 4=0;gto +0
6: dsp ""
7: wait 10+5e6/p4;rdi 4→p2
8: otdp2/p4→p2
9: wtb "MP";170040;mdec;ret
+3706
```

Bytes used: 296

HP-IB

```
0: "Time Interval":
1: "TI":gsb "Parameter Check"
2: 100000→p4
3: if p0<3;dsp "No return variable in cll statement";beep;end
4: wrt "MP.2","O";40+dtopp0,"T";char(64+p1),"T"
5: wrt "MP.2","O";240+dtopp0,"T";char(64+p1),"2"
6: dsp "Waiting for pulse";red "MP";p2;if p2=0;gto +0
7: dsp ""
8: wait 10+5e6/p4;red "MP";p2;otdp2/p4→p2
9: wrt "MP.1","040T";ret
+19515
```

Bytes used: 340

File 25: Frequency Measurement (FM)

Returns the decimal value of an external frequency source. The program adjusts the sample interval (from 69600B) until the best resolution is achieved.

SYNTAX

cll 'Frequency Measurement' (S_0, D_0, [U])

or

cll 'FM' (S_0, D_0, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1	Card Slot Address	None
D_n	p2	Storage Location for Frequency Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Frequency Measurement' (401, A, 0)

or

cll 'Frequency Measurement' (1, A)

or

cll 'FM' (401, A)

In the first example, the long descriptive label and the complete slot address (401) is used, the data storage location is specified, and the unit number is included. In the next example, the shortened slot address (1) is used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1 — Used for slot address of 69435A Pulse Counter card. The value must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot address must be in decimal.

p2 — Used to store the results of the frequency measurement. Data is in decimal representing the number of hertz.

pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.
p40	Sample time interval for the 69600B Programmable Timer card. Value is stepped from 10 to 100 to 1000 milliseconds by the subprogram.

PROGRAM DESCRIPTION

This subprogram requires two cards: the 69435A Pulse Counter and 69600B Programmable Timer card (see the connection diagram of Figure 3-4). The purpose of this subprogram is to measure the frequency (in hertz) of an external frequency source connected to the 69435A card.

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word, with SYE mode bit on, to the unit designated by pp0. The pulse counter card is then preset to zero and the 69600B Programmable Timer card programmed for a 10 millisecond pulse. The 69600B is assumed to be in the slot immediately to the right of the 69435A card. If the data read back from the 69435A is less than 144 octal (100 decimal) and if the gate time variable, p40, is less than 1000 milliseconds, the program will repeat the measurement. Repeating the measurement improves the resolution and accuracy of the results.

Once the reading from the 69435A has enough resolution, the data is converted to frequency. Gate time variable, p40, is converted to seconds (0.001 * p40) and the count, in decimal, is divided by the gate time. Before returning to the main program, the subprogram sends a control word, with SYE mode bit on, to unit zero, the 6940B.

SPECIAL CONSIDERATIONS

1. The 69600B must be located in the slot immediately to the right of the 69435A.
2. Connections are those given in the Frequency Measurement Connection Diagram.
3. The 69600B must be modified by the user to have jumper W6 out and jumper W3 in.
4. Unknown frequency source is capable of providing drive current (5-10 milliamps) for count inputs.

USED WITH

69435A Pulse Counter Card, 69600B Programmable Timer.

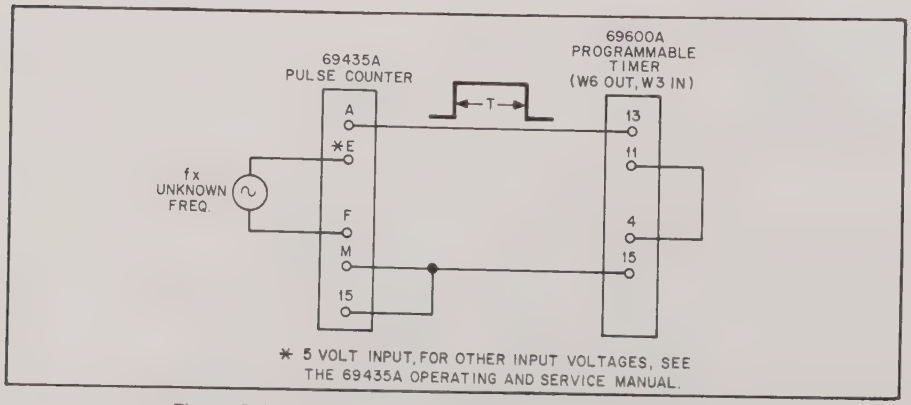


Figure 3-4. Frequency Measurement Connection Diagram

PROGRAM LISTINGS

GPIO

```
0: "Frequency Measurement":
1: "FM":esb "Parameter Check"
2: 10→p40
3: wtb "MP",170040+dtopp0,10000*dtop1
4: wtb "MP",170160+dtopp0,10000*(dtop1+1)+dtop40,170240
5: wti 4,10000*dtop1;rdi 4→p2
6: if p2<144 and p40<1000;10p40→p40;gto -3
7: otdp2/(.001*p40)→p2
8: wtb "MP",170040;mdec;ret
*20794
```

Bytes used: 292

HP-IB

```
0: "Frequency Measurement":
1: "FM":esb "Parameter Check"
2: 10→p40
3: wrt "MP.2","O",40+dtopp0,"T";char(64+p1),"T"
4: wrt "MP.2","O",160+dtopp0,"T",char(64+p1+1),dtop40,"T","O",240+dtopp0,"T"
5: wrt "MP",char(64+p1),"T";red 723,p2
6: if p2<144 and p40<1000;10p40→p40;gto -3
7: otdp2/(.001*p40)→p2
8: wrt "MP.1","040T";ret
*2300
```

Bytes used: 336

File 26: Option 40 Voltage Control (VC)

Sends desired output voltage (0 to 100V) in decimal form to the option 40 power supply programming cards specified.

SYNTAX

cll 'Option 40 Voltage Control' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'VC' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Option 40 Voltage Control' (401, A, 407, G, 402, B, 0)

or

cll 'Option 40 Voltage Control' (1, 10, 7, 20, 2, 30)

or

cll 'VC' (401, A, 407, 20, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are data to be sent to output cards. The data is in decimal (0 to 100) and represents the desired output voltage in volts.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.
p38	Power supply LSB: 0.01 for power supplies of up to 40 volts output. 0.025 for power supplies of 41 to 100 volts maximum output.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The data (voltages) are converted to LSB's and programmed to the resistance programming card specified. For power supplies in the range of 0 to 40 volts, the LSB is 0.01 volts. For power supplies in the range of 41 to 100 volts, the LSB is 0.025 volts.

This program waits until each resistance programming card completes before programming the next card. When all the cards have been programmed, this routine turns on SYE at the Multiprogrammer and returns to the main program.

SPECIAL CONSIDERATIONS

The proper resistance programming card must be used to program the power supply being used (see Table 3-1).

USED WITH

69500A to 69506A Resistance Output Cards.

PROGRAM LISTINGS

GPIO

```
0: "Option 40 Voltage Control":
1: "VC":gsb "Parameter Check"
2: wtb "MP",170160+dtopp0
3: 1→p39
4: if p(p39+1)>-.001 and p(p39+1)<100;gto +2
5: beep;prt "Data p",p39+1,"out of range";stp ;gto -1
6: if p(p39+1)<40;.025→p38;gto +2
7: .01→p38
8: dtop(p39+1)/p38→p39
9: wtb "MP",10000dtop p39+p38
10: if (p39+2→p39)<=p0-1;gto -6
11: wtb "MP",170040;rdec
12: ret
*993
```

Bytes used: 362

File 26: Option 40 Voltage Control (VC)

HP-IB

```
0: "Option 40 Voltage Control":
1: "VC":gsb "Parameter Check"
2: wrt "MP.2","O";160+dtop@0,"T"
3: 1→@39
4: if @(@39+1)>-.001 and @(@39+1)<100;gto +2
5: beep;prt "Data @",@39+1,"out of range";stp ;gto -1
6: if @(@39+1)>40;.025→@38;gto +2
7: .01→@38
8: dtop(@39+1)/@38→@38
9: wrt "MP.2",char(64+@@39),@38,"T"
10: if (@39+2→@39)<=@0-1;gto -6
11: wrt "MP.1","O40T"
12: ret
*25715
```

Bytes used: 374

Table 3-1.
HP Power Supply (Equipped with Option 040) and Resistance Output Card Selection Table

HP POWER SUPPLY			VOLTAGE PROGRAMMING			CURRENT PROGRAMMING §		
Volts	Amps	Model	Resistance Card	Volt/Step	Accuracy	Resistance Card	Amp/Step	Accuracy ‡
0-7.5	0-5	6281A	69501A	10mV	1% ± 20mV	69501A	10mA	10% ± 65mA
0-8	0-1000	6464C	69501A	10mV	1.0% ± 5mV	69501A	2A	1.0% ± 2A
0-10	0-2	6113A	69504A	10mV	0.1% ± 2mV	69512A	40mA	◇
0-10	0-10	6282A	69501A	10mV	1% ± 20mV	69501A	20mA	10% ± 65mA
0-10	0-20	6256B	69501A	10mV	1.0% ± 10mV	69510A	0.4A	10% ± 85mA
0-10	0-50	6259B	69501A	10mV	1.0% ± 8mV	69510A	1A	10% ± 65mA
0-10	0-100	6260B	69501A	10mV	1.0% ± 10mV	69510A	2A	10% ± 205mA
0-16 or 0-18	0-600 or 0-500	6466C	69501A	10mV	1.0% ± 5mV	69501A	3A	1.0% ± 2A
0-20 0-40 Dual Output; Dual Range	0-0.6 0-0.3	6205B	2 Each 69501A	10mV	1% ± 20mV	Current Limit Not Adjustable		
0-20	0-1	6101A	69504A	10mV	0.1% ± 2mV	69512A	20mA	◇
0-20	0-1	6111A †	69504A	10mV	0.1% ± 2mV	69512A	20mA	◇
0-20 0-40	0-2 0-1	6104A †† 6114A ††	69505A	10mV	.025% ± 1mV	69512A	40mA	2% ± 5mA
0-20	0-3	6284A	69501A	10mV	1% ± 20mV	69513A	60mA	10% ± 60mA
0-20	0-3	6253A *	2 Each 69501A	10mV	1% ± 20mV	69513A	60mA	10% ± 60mA
0-20	0-10	6263B	69501A	10mV	1.0% ± 9mV	69512A	0.2A	10% ± 50mA
0-20	0-20	6264B	69501A	10mV	1.0% ± 10mV	69510A	0.4A	10% ± 85mA
0-20	0-50	6261B	69501A	10mV	1.0% ± 8mV	69510A	1.0A	10% ± 65mA
0-24	0-3	6224B	69501A	10mV	1.0% ± 15mV	69513A	60mA	10% ± 35mA
0-25	0-1	6220B	69502A	25mV	1% ± 50mV	69512A	20mA	10% ± 20mA
0-25	0-2	6227B *	2 Each 69501A	10mV	1% ± 25mV	69512A	40mA	10% ± 15mA
0-36	0-300	6469C	69501A	10mV	1.0% ± 5mV	69501A	0.6A	1.0% ± 1A
0-40	0-0.5	6102A	69504A	10mV	0.1% ± 2mV	69512A	10mA	◇
0-40	0-0.5	6112A †	69504A	10mV	0.1% ± 2mV	69512A	10mA	◇
0-40	0-1.5	6289A	69501A	10mV	1% ± 20mV	69511A	30mA	10% ± 30mA
0-40	0-1.5	6255A*	2 Each 69501A	10mV	1% ± 20mV	69511A	30mA	10% ± 30mA
0-40	0-3	6265B	69501A	10mV	1.0% ± 7mV	69512A	60mA	10% ± 12mA
0-40	0-5	6266B	69501A	10mV	1.0% ± 8mV	69512A	0.1A	10% ± 20mA
0-40	0-10	6267B	69501A	10mV	1.0% ± 8mV	69512A	0.2A	10% ± 30mA
0-40	0-30	6268B	69501A	10mV	1.0% ± 8mV	69510A	0.60A	10% ± 40mA
0-40	0-50	6269B	69501A	10mV	1.0% ± 8mV	69510A	1.0A	10% ± 65mA
0-50	0-0.5	6220B	69502A	25mV	1% ± 25mV	69512A	10mA	10% ± 10mA
0-50 0-100	0-0.8 0-0.4	6105A †† 6115A ††	69506A	25mV	.025% ± 1mV	69511A	15mA	2% ± 5mA
0-50	0-1	6228B *	2 Each 69502A	25mV	1% ± 25mV	69512A	20mA	10% ± 15mA
0-50	0-1.5	6226B	69502A	25mV	1.0% ± 15mV	69511A	30mA	10% ± 18mA
0-60	0-1	6294A	69502A	25mV	1% ± 50mV	69512A	20mA	10% ± 20mA
0-60	0-3	6271B	69502A	25mV	1.0% ± 6mV	69512A	60mA	10% ± 12mA
0-60	0-15	6274B	69502A	25mV	1.0% ± 7mV	69512A	0.3A	10% ± 50mA
0-64	0-150	6472C	69502A	25mV	1.0% ± 25mV	69501A	0.3A	1.0% ± 1A
0-100	0-0.2	6106A	69506A	50mV	.1% ± 5mV	69512A	4mA	
0-100	0-0.75	6299A	69502A	25mV	1% ± 15mV	69511A	15mA	10% ± 15mA

§ Each current programming card (models 69510A-69513A) provides two independent outputs, and thus can program the current output of two power supplies. The 69501A card has only one output and is used for current programming of certain high current supplies.

‡ The accuracy specification refers to the combination of the power supply and the programming card; it includes the effects of load related internal temperature changes in the supply.
These supplies are constant voltage/current limit, rather than constant voltage/constant current.

† The 6111A and 6112A supplies are identical to the 6101A and 6102A supplies respectively, except for the addition of front panel thumbwheel voltage controls.

†† No special option 040 modifications are necessary for these dual range 6104A, 6105A, 6114A, and 6115A.

* Dual Output.

File 27: Option 40 Current Control (CC)

Sends desired current output (0 to 1.000 amps) in decimal form (e.g., 0.55 for 0.55A) to the option 40 power supply programming cards specified. Each card has two output channels (see program examples).

SYNTAX

cll 'Option 40 Current Control ' (S_0, D_0, S_0, D_1, S_1, D_2, S_1, D_3, - - - - S_n, D_n, [U])

or

cll 'CC' (S_0, D_0, S_0, D_1, S_1, D_2, S_1, D_3, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Option 40 Current Control' (401, A, 401, G, 407, B, 407, C, 0)

or

cll 'Option 40 Current Control' (1, .100, 1, .200, 7, .300, 7, .400)

or

cll 'CC' (401, A, 401, .200, 407, B, 407, C)

In the first example, the long descriptive label and the complete slot addresses (401, 407) are used, variable data is sent, and the unit number is included. Note that each slot address is repeated. The first data item (A) is used to program channel 1 of the card in slot 401. The second data item (G) is for channel 2 of the card in slot 401. The third data item (B) programs channel 1 of the next card specified, and so forth. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.	All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal and are repeated (see description below).
p2, p4, etc.	All even number variables are data to be sent to output cards. The data is in decimal (0.000 to 1.000) and represents the desired output current in amperes. The parameters are arranged in sets of four. First the card slot address, then the data to program channel 1, followed by a repeat of the card slot address and then the data to program channel 2.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The data (currents) are converted to LSB's and programmed to the resistance programming card specified.

The routine as written is intended for 1 amp supplies. A change must be made to program power supplies of other output ratings. Lines 4 and 8 contain the LSB of the power supply. For 1 amp power supplies this LSB is 0.02. For other power supplies, the "1" on lines 4 and 8 should be changed to the maximum rated output current. For example, to program a 2 amp supply, the "1" on lines 4 and 8 should be changed to a "2".

This program waits until each card has completed before programming the next card. When all cards have been programmed, this routine turns on SYE at the multiprogrammer and returns to the main program.

SPECIAL CONSIDERATIONS

1. The proper resistance programming card must be used to program the power supply being used (see Table 3-1).
2. Lines 4 and 8 must be modified if the supply is not rated for one amp: the "1" in these lines should be changed to correspond to the maximum output current of the particular supply.
3. The resistance cards are programmed by a pair of data: S_0, D_1, S_0, D_2. The first Address-Data pair for chanel one, the second pair for channel 2. This requires that each address be repeated twice: once for the channel one of the card and the second time for channel two. If only one channel of a card is to be programmed, a zero must be programmed to the other channel.

USED WITH

69510A to 69351A Resistance Output Cards.

PROGRAM LISTINGS

GPIO

```
0: "Option 40 Current Control":
1: "CC":gsb "Parameter Check"
2: wtb "MP";170160+dtopp0
3: 1→p39
4: p(p39+1)/(1*.02)→p38
5: if p(p39+1)>-.001 and p(p39+1)<50;gto +2
6: beep;prt "Data p";p39+1;"out of range";stp ;gto -2
7: if p39+3>p0;0→p37;gto +4
8: p(p39+3)/(1*.02)→p37
9: if p37>-.001 and p37<=50;gto +2
10: beep;prt "Data p";p39+3;"out of range";stp ;gto -2
11: wtb "MP";10000dtopp39+dtop38+100dtop37
12: if (p39+4→p39)<=p0-1;gto -8
13: wtb "MP";170040;mdec
14: ret
+12388
```

Bytes used: 470

HP-IB

```
0: "Option 40 Current Control":
1: "CC":gsb "Parameter Check"
2: wrt "MP.2","O";160+dtopp0;"T"
3: 1→p39
4: p(p39+1)/(1*.02)→p38
5: if p(p39+1)>-.001 and p(p39+1)<50;gto +2
6: beep;prt "Data p";p39+1;"out of range";stp ;gto -2
7: if p39+3>p0;0→p37;gto +4
8: p(p39+3)/(1*.02)→p37
9: if p37>-.001 and p37<=50;gto +2
10: beep;prt "Data p";p39+3;"out of range";stp ;gto -2
11: wrt "MP.1";char(64+pp39);dtop38+100dtop37;"T"
12: if (p39+4→p39)<=p0-1;gto -8
13: wrt "MP.1","O40T"
14: ret
+17510
```

Bytes used: 482

File 28: Arm (AM)

Arms cards specified to permit an interrupt whenever any card detects an interrupt condition. (Assumes all W6 jumpers have been removed.)

SYNTAX

cll 'Arm' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'AM' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Dummy Variables	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Arm' (401, 0, 407, 0, 402, 0, 0)

or

cll 'Arm' (1, 0, 7, 0, 2, 0)

or

cll 'AM' (401, 0, 407, 0, 402, 0)

In the first example, the long descriptive label is used, the complete slot addresses (401, 407, etc.) of the cards to be armed are specified, dummy data values (zeros) are included, and the unit number is given. A dummy data value in the range from 0 to 7777 must be included after each slot address. If the dummy data is omitted, the specified cards will not be armed.

In the next example, shortened slot addresses (1, 7, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.	All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.
p2, p4, etc.	All even number variables are dummy variables, and must be in the range of zero through 7777.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The control word programs the ISL and SYE mode bits on. All cards specified are addressed and gated. After the last card is gated, a control word is sent with IEN, SYE and TME mode bits on. This control word places all 6940B/6941B mainframes in the "Interrupt Mode". All cards selectively armed with this call, are then capable of generating an interrupt which the 9825A can detect. The program then returns to the main program, after enabling the interrupt system of the 9825A (eir "MP"). Figures 3-5 (page 3-90) and 3-6 (page 3-91) are interrupt mode flow charts that illustrate the programming sequence for the "Arm" subprogram.

SPECIAL CONSIDERATIONS

1. Dummy data variables (0 to 7777) must be used in this subprogram.
2. These cards will become armed, unless jumper W6 is removed from all, when this call sends a control word with IEN, SYE and TME mode bits on.
3. Cards previously addressed when TME was off will generate an interrupt if they have not completed their time out prior to this call.

USED WITH

69431A Digital Input, 69434A Event Sense, 69436A Process Interrupt, 69600B Programmable Timer.

PROGRAM LISTINGS

GPIO

```
0: "Arm":
1: "AM":esb "Parameter Check"
2: wtb "MP",170240+dtopp0
3: 1→p39
4: wtb "MP",10000*dtopp39
5: if (p39+2→p39)<=p0-1;gto -1
6: wtb "MP",170460;mdec
7: eir "MP";ret
#4661
```

Bytes used: 166

HP-IB

```
0: "Arm":
1: "AM":esb "Parameter Check"
2: wrt "MP.2","O",240+dtopp0,"T"
3: 1→p39
4: wrt "MP.1",chr(pp39+64),"T"
5: if (p39+2→p39)<=p0-1;gto -1
6: wrt "MP.1","00460T"
7: eir "HP-IB";ret
#20408
```

Bytes used: 184

Performs a Serial Poll on cards specified. The data returned is in octal and includes an IRQ bit if the card generated an interrupt.

SYNTAX

cll 'Serial Poll' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'SP' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Serial Poll' (401, A, 407, G, 402, B, 0)

or

cll 'Serial Poll' (1, A, 7, G, 2, B)

or

cll 'SP' (401, A, 407, G, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, the data storage locations are specified, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even number variables are used to store data from input cards. The IRQ bit is not removed in the call subroutine. Data will be returned in octal.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word, with ISL and SYE mode bits on, to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The program then addresses and reads (without a gate) all cards specified. The data, as well as any IRQ bits, are stored in the variables specified. When present, the IRQ bit indicates that the card interrupted. After the last card is polled, a control word with SYE mode bit on is sent to the unit zero. The program then returns to the main program. Figures 3-5 (page 3-90) and 3-6 (page 3-91) are interrupt mode flow charts that illustrate the programming sequence for the "Serial Poll" subprogram.

SPECIAL CONSIDERATIONS

The user program must be written to determine which return data contains an input request (IRQ) bit and therefore caused an interrupt. The following is a sample program using the interrupt mode subprograms.

```
0: oni 9;"ISR"
1: cll 'Recycle'(1;0;2;0)
2: cll 'Arm'(1;0;2;0)
                •
                •
                •
13: sto +0
14: "ISR":
15: cll 'Serial Poll'(1;A;2;B)
16: if A>7777;sto "N"
17: if B>7777;sto "P"
18: iret
19: "N":cll 'Disarm'(1;0)
20: prt A-100000
21: iret
22: "P":cll 'Disarm'(2;0)
23: prt B-100000
24: iret
```

For HP-IB change lines 0, 20, and 23 to:

0: oni 7, "ISR"; rds (723)→C
20: prt (A−10000)
23: prt (B−10000)

USED WITH

69431A Digital Input, 69434A Event Sense, 69436A Proccess Interrupt, 69600B Programmable Timer.

PROGRAM LISTINGS

GPIO

```
0: "Serial Poll":
1: "SP":gsb "Parameter Check"
2: wtb "MP";170240+dtopp0
3: 1→p39
4: wti 4;10000+dtopp39
5: rdi 4→p(p39+1)
6: if (p39+2→p39)<=p0-1;gto -2
7: wtb "MP";170040;mdec
8: ret
*3710
```

Bytes used: 186

HP-IB

```
0: "Serial Poll":
1: "SP":gsb "Parameter Check"
2: wrt "MP.2","O",240+dtopp0,"T"
3: 1→p39
4: wrt "MP.1",char(pp39+64),"X"
5: red "MP",p(p39+1)
6: if (p39+2→p39)<=p0-1;gto -2
7: wrt "MP.1","00040T"
8: ret
*28995
```

Bytes used: 202

Selectively disarms cards specified. Assumes all W6 jumpers have been removed.

SYNTAX

cll 'Disarm' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'DM' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Dummy Variables	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Disarm' (401, 0, 407, 0, 402, 0, 0)

or

cll 'Disarm' (1, 0, 7, 0, 2, 0)

or

cll 'DM' (401, 0, 407, 0, 402, 0)

In the first example, the long descriptive label is used, the complete slot addresses (401, 407, etc.) of the cards to be disarmed are specified, dummy data values (zeros) are included, and the unit number is given. A dummy data value in the range from 0 to 7777 must be included after each slot address. If the dummy data is omitted, the specified cards will not be disarmed.

In the next example, shortened slot addresses (1, 7, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.	All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.
p2, p4, etc.	All even number variables are dummy variables, and must be between zero and 7777.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The control word program the SYE mode bit on. All cards specified are sent zero data. The data is sent with a gate signal. A return to the main program is made after the last card is disarmed and a control word with SYE mode bit on is sent to unit zero, the 6940B. Figures 3-5 (page 3-90) and 3-6 (page 3-91) are interrupt mode flow charts that illustrate the programming sequence for the "Disarm" subprogram.

SPECIAL CONSIDERATIONS

1. Removal of jumper W6 from 69431A, 69434A, 69436A and 69600B cards is recommended. These cards will become armed when a control word with IEN, SYE and TME mode bits on is programmed, unless their W6 jumpers are removed.

USED WITH

69431A Digital Iput, 69434A Event Sense, 69436A Process Interrupt, 69600B Programmable Timer.

PROGRAM LISTINGS

GPIO

```
0: "Disarm":
1: "DM":ssb "Parameter Check"
2: wtb "MP";170040+dtopp0
3: 1→p39
4: wtb "MP";10000*dtopp39
5: if (p39+2→p39)<=p0-1;gto -1
6: wtb "MP";170040
7: ret
*4890
```

Bytes used: 158

HP-IB

```
0: "Disarm":
1: "DM":ssb "Parameter Check"
2: wrt "MP.2";"O";40+dtopp0;"T"
3: 1→p39
4: wrt "MP.1";char(pp39+64);"T"
5: if (p39+2→p39)<=p0-1;gto -1
6: wrt "MP.1";"00040T"
7: ret
*31345
```

Bytes used: 176

Halts system until all output cards complete their operations.

SYNTAX

cll 'Time Out'

or

cll 'TO'

PARAMETERS

None required.

PROGRAM EXAMPLES

cll 'Time Out'

or

cll 'TO'

The first example uses the long descriptive title to call 'Time Out'. The second example uses the short label.

PROGRAM DESCRIPTION

The subprogram calls 'Parameter Check' to set up formats and modes necessary to "talk" to the Multiprogrammer. The subprogram then sends out a control word to the Multiprogrammer with TME, DTE, and SYE on. The system will now await any card, such as the Stepping Motor Card, that is in the middle of an operation to complete. When all cards finish their previously programmed operation, this subprogram turns SYE on and returns.

SPECIAL CONSIDERATIONS

1. Any card timing out in any mainframe will cause the system to wait when this subprogram is called.
2. This subprogram will work especially well if the user wishes to wait while a stepping motor is turning. First call 'Motor Output', then call 'Time Out' to wait for the motor to stop.

USED WITH

All output cards.

PROGRAM LISTINGS

GPIO

```
0: "Time Out":
1: "TO":cll 'PC'(0)
2: wtb "MP",170160,170040
3: mdec;ret
*24250
```

Bytes used: 68

HP-IB

```
0: "Time Out":
1: "TO":cll 'PC'(0)
2: wrt "MP.1","00160T00040T"
3: ret
*18026
```

Bytes used: 68

Recycles specified cards, before they are armed by the subprogram 'ARM' (File 28).

SYNTAX

cll 'Recycle' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'RE' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Card Data In Octal	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Recycle' (401, A, 407, G, 402, B, 0)

or

cll 'Recycle' (1, 5252, 7, 2525, 2, 7777)

or

cll 'RE' (401, A, 407, 77, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, variable data is sent, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent,and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, complete slot addresses are used, variable data and constant data are sent, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.	All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.
p2, p4, etc.	All even number variables are data to be sent to the cards. If these variables are greater than 7777, the program will stop and permit entry of a new value. Data must be in octal.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The data parameters represent the following:

69436A: Data is decimal representation of those bits to be cleared

69600B : Data is decimal representation of next time period. Note that the "Timer Pulse" subprogram may be used to recycle the 69600B.

SPECIAL CONSIDERATIONS

1. Removal of jumper W6 is recommended to enable individual arming of cards. See the proper Operating and Service Manual for the W6 location on the card.
2. At start up the 69436A must be initialized to reset any bits set on power up. To accomplish this, data of 4095 may be sent to all 69436A cards using this subprogram: "Recycle"

USED WITH

69436A Process Interrupt Card, 69600B Programmable Timer Card.

DO NOT USE WITH

69332A Open Collector Output, 69380A Breadboard Output, and 69601B Frequency Reference.

PROGRAM LISTINGS

GPIO

```
0: "Recycle":
1: "RE":gsb "Parameter Check"
2: wtb "MP",170040+dtopp0
3: 1→p39
4: wtb "MP",10000*dtopp39+dtop(p39+1)
5: if (p39+2→p39)<=p0-1;gto -1
6: wtb "MP",170040;ndec
7: ret
+16432
```

Bytes used: 176

HP-IB

```
0: "Recycle":
1: "RE":gsb "Parameter Check"
2: fmt 1,c,zifmt 2,c,f4.0,c,z;wrt "MP.2","0",40+dtopp0,"T"
3: 1→p39
4: wrt "MP.2",char(pp39+64),dtop(p39+1),"T"
5: if (p39+2→p39)<=p0-1;gto -1
6: wrt "MP.1","00040T"
7: ret
+11538
```

Bytes used: 216

File 33: Reference Word (RW)

Reads back the reference words from the desired Event Sense cards and sends new reference words to these cards.

SYNTAX

cll 'Reference Word' (S_0, D_0, S_0, D_1, S_1, D_2, S_1, D_3, - - - - S_n, D_n, [U])

or

cll 'RW' (S_0, D_0, S_0, D_1, D_2, S_1, D_3, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Reference Word	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Reference Word' (401, A, 401, G, 407, B, 407, C, 0)

or

cll 'Reference Word' (1, A, 1, 7777, 7, B, 7, 5252)

or

cll 'RW' (401, A, 401, G, 407, B, 407, C)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, and the unit number is included. Note that each slot address is repeated. The first data item (A) is the storage location which will receive the reference word read from the card in slot 401. The second data item (G) programs the new reference word to the card in slot 401. In the next example, shortened slot addresses (1, 7, etc.) are used, constant data is sent, for the new reference words, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The sub-program will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.	All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal and are repeated (see description below).
p2, p4, etc.	All even numbered parameters are data which represent the reference word in octal. The parameters are arranged in sets of four: First the card slot address, then the variable to receive the reference word read from the card, followed by a repeat of the card slot address and then the data to program the new reference word. If it is desired to retain the old reference word, the variable which receives the old reference word may be used to program the new one as: cll 'RW' (1, A, 1, A) In this example "A" will receive the old reference word, which will be returned to the card.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The first data parameter receives the old reference word, the second data parameter will become the card's new reference word. These can both be the same if the same variables are repeated in both locations.

If it is desired to change the reference word to equal the external word, the following procedure should be followed.

(1) Read the card using "Serial Poll" as in: cll 'SP' (1,A)
(2) Call 'Reference Word as in: cll 'RW' (1, X, 1, A-IRQ)
where, X is a dummy variable to receive the old reference word, 1 is the card address, A is the external word, and "IRQ" is used to subtract IRQ (HP-IB: IRQ = 10000, GPIO: IRQ = 100000)

Note that the external word is lost once "Reference Word" has been called. Figure 3-6 (page 3-91) is an interrupt mode flow chart that illustrates the programming sequence for the Reference Word subprogram.

SPECIAL CONSIDERATIONS

The "Reference Word" subprogram uses two address-data pairs for each card as: "cll 'RW' (1, A, 1, B)" where "1" is the card address, "A" is the variable to receive the old reference word and "B" is the new reference word to be programmed.

USED WITH

69434A Event Sense Card.

PROGRAM LISTINGS

GPIO

```
0: "Reference Word":
1: "RW":ssb "Parameter Check"
2: 1→p39
3: wtb "MP",170240+dtopp0;wti 4,10000*dtopp39
4: rdb("MP")→p(p39+1)
5: if p(p39+1)>7777;p(p39+1)-100000→p(p39+1)
6: if p39+3>p0;sto +2
7: wtb "MP",170040+dtopp0,10000*dtop(p39+2)+p(p39+3)
8: if (p39+4→p39)<p0-1;sto -5
9: wtb "MP",170040;mdec;ret
*8492
```

Bytes used: 306

HP-IB

```
0: "Reference Word":
1: "RW":ssb "Parameter Check"
2: 1→p39
3: wrt "MP.2","O",240+dtopp0,"T",char(64+pp39),"X"
4: red "MP",p(p39+1)
5: if p(p39+1)>7777;p(p39+1)-10000→p(p39+1)
6: if p39+3>p0;sto +2
7: wrt "MP.2","O",40+dtopp0,"T",char(p(p39+2)),p(p39+3),"T"
8: if (p39+4→p39)<p0-1;sto -5
9: wrt "MP.1","O0040T";ret
*17881
```

Bytes used: 322

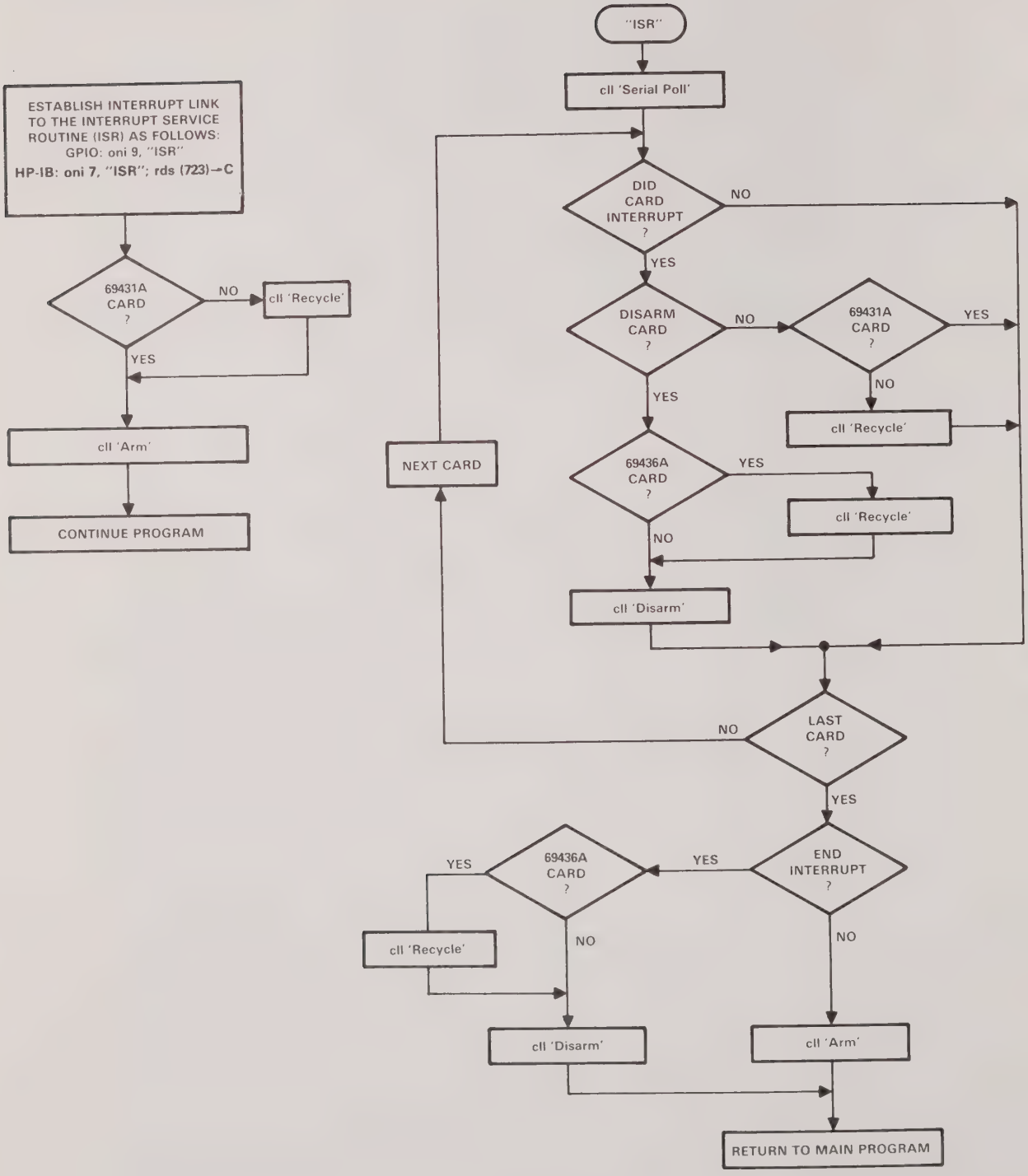


Figure 3-5. Interrupt Mode Flow Chart for the 69431A Digital Input, 69436A Process Interrupt, and 69600B Programmable Timer Cards.

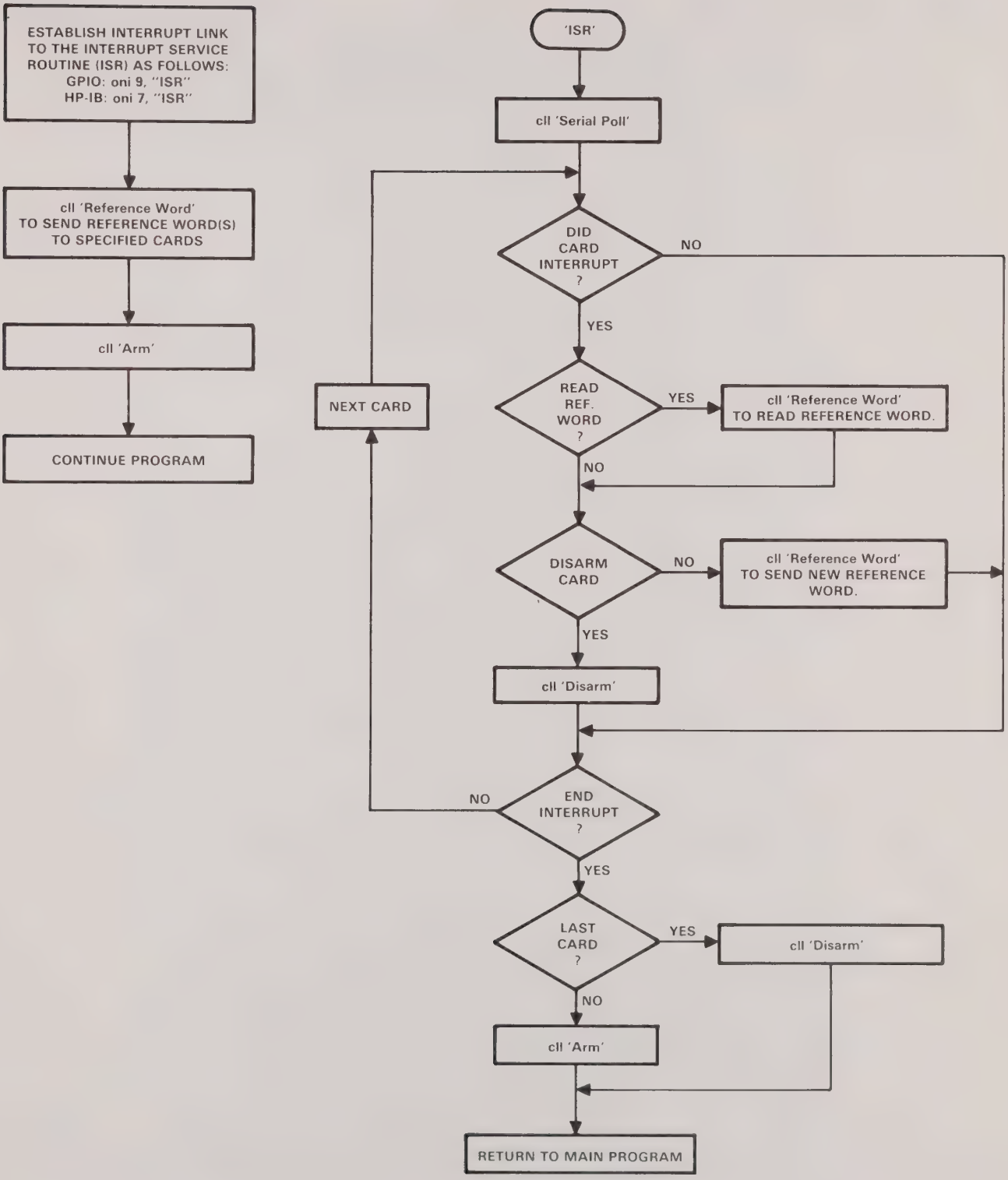


Figure 3-6. Interrupt Mode Flow Chart for the 69434A Event Sense Card.

File 34: T Thermocouple (TT)

Reads the temperature in °C from 69423A cards and type T thermocouples (copper versus copper-nickel alloy). Each 69423A card has six input channels (see program examples.)

SYNTAX

cll 'T Thermocouple' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'TT' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot and Channel Address	None
D_n	p2, p4, etc.	Storage Locations for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'T Thermocouple' (401.2, A, 401.3, G, 402.5, B, 0)

or

cll 'T Thermocouple' (1.2, A, 1.3, G, 2.5, B)

or

cll 'TT' (1.2, A, 1.3, G, 2.5, B)

In the first example, the long descriptive label and the complete slot addresses (401.2, 401.3, etc.) are used, the data storage locations are specified, and the unit number is included. Note that the number to the right of the decimal point specifies the input channel (see variable description). In the next example, shortened slot addresses (1.2, 1.3, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses and channel numbers. The slot address is expressed as a slot number to the left of the decimal point and a channel to the right. The slot address values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. The channel numbers run from 1 to 6. Thus, in order to read channel

5 of the 69423A card in slot 2 use: "402.5" or "2.5". If the slot address or channel numbers are not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even numbered variables are used to store data from the input cards.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

p60-62 — Constants to convert from reference junction voltage to °C.

p50-53 — Constants to convert from reference junction °C to correction voltage of type T thermocouples.

p40-44 — Constants to convert from type T thermocouple voltage to °C.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram first reads the reference junction voltage and converts this to °C. This value in °C is then converted to a correction voltage for type T thermocouples. The specified channel of the addressed 69423A card is then read and converted to °C.

The conversion equation from °C of reference temperature to voltage correction factor is of the form:

$p50 + X (p51 + X (p52 + X p53)) = \mu V$

where, $p50 = a_0$, $p51 = a_1$, $p52 = a_2$, $p53 = a_3$.

The conversion equation from volts from the thermocouple junction to °C is of the form:

$p40 + X (p41 + X (p42 + X (p43 + p44))) = °C$

where, $p40 = a_0$, $p41 = a_1$, $p42 = a_2$, $p43 = a_3$, $p44 = a_4$.

The constants are taken from the Thermocouple Reference Tables Based on the IPTS — 68, U. S. Department of Commerce (March 1974). The constants chosen for this subprogram are for the following temperature range:

Temperature Range	Error Range
0-400°C	−.15 to .17°C

For other ranges, change the p40-p44 constants to the values given in Table 3-2 for the applicable temperature range.

Table 3-2. Type T Thermocouples — Quartic Approximations to the data as a function of voltage in selected temperature ranges (°C). The expansion is of the form $T = a_0, a_1E, + a_2E^2 + a_3E^3 + a_4E^4$ where E is in microvolts and T is in degrees Celsius.

Temperature Range (°C)	a_0		a_1		a_2		a_3		a_4		Error Range (°C)
	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Error-Approx.
-270 to 0	. . . . .	. .	4. 3553379	-3	-2. 0325426	-5	-5. 4720813	-9	-5. 0865527	-13	-.8 to .6
-200 to 0	. . . . .	. .	2. 3837090	-2	-2. 9878839	-6	-7. 1945810	-10	-1. 0041943	-13	-.3 to .3
-200 to 400	. . . . .	. .	2. 6792411	-2	-1. 0370271	-6	6. 1330327	-11	-1. 3988385	-15	-.6 to .5
0 to 400	. . . . .	. .	2. 5661297	-2	-6. 1954869	-7	2. 2181644	-11	-3. 5500900	-16	-.15 to .17

SPECIAL CONSIDERATIONS

This subprogram returns temperature in °C. To convert to °F, use the following equation:

$$9/5 \times C + 32 = F$$

USED WITH

69423A card and type T thermocouples (Copper vs. Copper-Nickel Alloy)

PROGRAM LISTINGS

GPIO

```
0: "T Thermocouple":
1: "TT":esb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 38.709457→p51;3.7085566e-2→p52;5.649552e-5→p53
4: 2.5661297e-2→p41;-6.1954869e-7→p42;2.2181644e-11→p43;-3.55009e-16→p44
5: 1→p39
6: wtb "MP",170160+dtopp0
7: if (10frc(pp39)→p38)<7;sto +2
8: beep;prt "Address p";p39;"out of range";stp ;sto -1
9: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
10: wti 4,10000dtoint(pp39);rdi 4→p37
11: (otdp37-4096)*10/4352→p37
12: p60/(p61+ln(-p37))-p62→p37
13: p37(p51+p37(p52+p37p53))1e-6→p37
14: wtb "MP";170160+dtopp0;10000dtoint(pp39)+p38;170260+dtopp0
15: wti 4;10000dtoint(pp39);rdi 4→p36
16: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
17: (p37+p36)1e6→p36
18: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
19: if (p39+2→p39)<=p0-1;sto -13
20: wtb "MP";170040;ndec
21: ret
+18707
```

Bytes used: 810

HP-IB

```
0: "T Thermocouple":
1: "TT":esb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 38.709457→p51;3.7085566e-2→p52;5.649552e-5→p53
4: 2.5661297e-2→p41;-6.1954869e-7→p42;2.2181644e-11→p43;-3.55009e-16→p44
5: 1→p39
6: wrt "MP.2","O",160+dtopp0,"T"
7: if (10frc(pp39)→p38)<7;sto +2
8: beep;prt "Address p";p39;"out of range";stp ;sto -1
9: wrt "MP.2";char(64+int(pp39)),7,"T","O",260+dtopp0,"T"
10: wrt "MP.1";char(64+int(pp39)),"X";red "MP",p37
11: (otdp37-4096)*10/4352→p37
12: p60/(p61+ln(-p37))-p62→p37
13: p37(p51+p37(p52+p37p53))1e-6→p37
14: wrt "MP.2","O",160+dtopp0,"T";char(int(pp39)+64),p38,"T"
15: wrt "MP.2","O",260+dtopp0,"T";char(64+int(pp39)),"X";red "MP",p36
16: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
17: (p37+p36)1e6→p36
18: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
19: if (p39+2→p39)<=p0-1;sto -13
20: wrt "MP.1","040T"
21: ret
+12725
```

Bytes used: 862

File 35: J Thermocouple (JT)

Reads the temperature in °C from 69423A cards and type J thermocouples (iron versus copper-nickel alloy). Each 69423A card has six input channels (see program examples.)

SYNTAX

cll 'J Thermocouple' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'JT' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot and Channel Address	None
D_n	p2, p4, etc.	Storage Locations for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'J Thermocouple' (401.2, A, 401.3, G, 402.5, B, 0)

or

cll 'J Thermocouple' (1.2, A, 1.3, G, 2.5, B)

or

cll 'JT' (1.2, A, 1.3, G, 2.5, B)

In the first example, the long descriptive label and the complete slot addresses (401.2, 401.3, etc.) are used, the data storage locations are specified, and the unit number is included. Note that the number to the right of the decimal point specifies the input channel (see variable description). In the next example, shortened slot addresses (1.2, 1.3, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses and channel numbers. The slot address is expressed as a slot number to the left of the decimal point and a channel to the right. The slot address values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. The channel numbers run from 1 to 6. Thus, in order to read channel 5 of the 69423A card in slot 2 use: "402.5" or "2.5". If the slot

address or channel numbers are not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even numbered variables are used to store data from the input cards.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

p60-62 — Constants to convert from reference junction voltage to °C.

p50-53 — Constants to convert from reference junction °C to correction voltage of type J thermocouples.

p40-44 — Constants to convert from type J thermocouple voltage to °C.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram first reads the reference junction voltage and converts this to °C. This value in °C is then converted to a correction voltage for type J thermocouples. The specified channel of the addressed 69423A card is then read and converted to °C.

The conversion equation from °C of reference temperature to voltage correction factor is of the form: μV

$p50 + X\ (p51 + X\ (p52 + X\ p53)) = mV$

where, $p50 = a_0$, $p51 = a_1$, $p52 = a_2$, $p53 = a_3$.

The conversion equation from volts from the thermocouple junction to °C is of the form:

$p40 + X\ (p41 + X\ (p42 + X\ (p43 + p44))) = °C$

where, $p40 = a_0$, $p41 = a_1$, $p42 = a_2$, $p43 = a_3$, $p44 = a_4$.

The constants are taken from the Thermocouple Reference Tables Based on the IPTS — 68, U. S. Department of Commerce (March 1974). The constants chosen for this subprogram are for the following temperature range:

Temperature Range	Error Range
−20 to 500°C	−.07 to .06°C

For other ranges, change the p40-p44 constants to the values given in Table 3-3 for the applicable temperature range.

Table 3-3. Type J Thermocouples — Quartic Approximations to the data as a function of voltage in selected temperature ranges (°C). The expansion is of the form $T = a_0, a_1E, + a_2E^2 + a_3E^3 + a_4E^4$ where E is in microvolts and T is in degrees Celsius.

Temperature Range (C°)	a_0		a_1		a_2		a_3		a_4		Error Range (C°)
	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Exact-Approx.
-200 to 0	. . . . .		1.8843850	-2	-1.2029733	-6	-2.5278593	-10	-2.5849263	-14	-.4 to .5
-200 to 760	. . . . .		2.1155170	-2	-3.3513149	-7	1.2443997	-11	-1.5227150	-16	-6 to 7
-200 to 1200	. . . . .		2.1676850	-2	-2.1844464	-7	3.9094347	-12	-2.4303017	-17	-14 to 10
-20 to 500	. . . . .		1.9745056	-2	-1.8094256	-7	7.8777919	-12	-1.1897222	-16	-.07 to .06
0 to 400	. . . . .		1.9750953	-2	-1.8542600	-7	8.3683958	-12	-1.3280568	-16	-.03 to .05
0 to 760	. . . . .		1.9323799	-2	-1.0306020	-7	3.7084018	-12	-5.1031937	-17	-.9 to .7
0 to 1200	. . . . .		1.8134974	-2	-5.6495930	-8	-2.4644023	-12	2.1141718	-17	-3 to 4
400 to 760	9.2808351	+1	5.4463817	-3	6.5254537	-7	-1.3987013	-11	9.9364476	-17	-.03 to .03
400 to 1200	-1.1075293	+2	2.8651303	-2	-2.9758175	-7	2.5945419	-12	-4.9012035	-18	-1.3 to 1.6
600 to 760	1.8020713	+2	-4.5284199	-3	1.0769294	-6	-2.1962321	-11	1.5521511	-16	-.001 to .001
760 to 1200	-6.3828680	+2	7.4068749	-2	-1.7177773	-6	2.1771293	-11	-9.9502571	-17	-.15 to .11

SPECIAL CONSIDERATIONS

This subprogram returns temperature in °C. To convert to °F, use the following equation:

$$9/5 \times C + 32 = F$$

USED WITH

69423A card and type J thermocouples (Iron vs. Copper-Nickel Alloy)

PROGRAM LISTINGS

GPIO

```
0: "J Thermocouple":
1: "JT":gsb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 50.373743→p51;3.0167011e-2→p52;-7.4293513e-5→p53
4: 1.97450056e-2→p41;-1.8094256e-7→p42;7.8777919e-12→p43;-1.1897222e-16→p44
5: 1→p39
6: wtb "MP",170160+dtopp0
7: if (10frc(pp39)→p38)<7;gto +2
8: beep;prt "Address p",p39,"out of range";stp ;gto -1
9: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
10: wti 4,10000dtoint(pp39);rdi 4→p37
11: (otdp37-4096)*10/4352→p37
12: p60/(p61+ln(-p37))-p62→p37
13: p37(p51+p37(p52+p37p53))1e-6→p37
14: wtb "MP",170160+dtopp0,10000dtoint(pp39)+p38,170260+dtopp0
15: wti 4,10000dtoint(pp39);rdi 4→p36
16: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
```

Bytes used: 814

HP-IB

```
0: "J Thermocouple":
1: "JT":gsb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 50.373743→p51;3.0167011e-2→p52;-7.4293513e-5→p53
4: 1.9745056e-2→p41;-1.8094256e-7→p42;7.8777919e-12→p43;-1.1897222e-16→p44
5: 1→p39
6: wrt "MP.2","O",160+dtopp0,"T"
7: if (10frc(pp39)→p38)<7;gto +2
8: beep;prt "Address p",p39,"out of range";stp ;gto -1
9: wrt "MP.2",char(64+int(pp39)),7,"T","O",260+dtopp0,"T"
10: wrt "MP.1",char(64+int(pp39)),"X";red "MP",p37
11: (otdp37-4096)*10/4352→p37
12: p60/(p61+ln(-p37))-p62→p37
13: p37(p51+p37(p52+p37p53))1e-6→p37
14: wrt "MP.2","O",160+dtopp0,"T",char(int(pp39)+64),p38,"T"
15: wrt "MP.2","O",260+dtopp0,"T",char(64+int(pp39)),"X";red "MP",p36
16: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
17: (p37+p36)1e6→p36
18: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
```

Bytes used: 864

File 36: E Thermocouple (ET)

Reads the temperature in °C from 69423A cards and type E thermocouples (nickel-chromium alloy versus copper-nickel alloy). Each 69423A card has six input channels (see program examples.)

SYNTAX

cll 'E Thermocouple' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'ET' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot and Channel Address	None
D_n	p2, p4, etc.	Storage Locations for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'E Thermocouple' (401.2, A, 401.3, G, 402.5, B, 0)

or

cll 'E Thermocouple' (1.2, A, 1.3, G, 2.5, B)

or

cll 'ET' (1.2, A, 1.3, G, 2.5, B)

In the first example, the long descriptive label and the complete slot addresses (401.2, 401.3, etc.) are used, the data storage locations are specified, and the unit number is included. Note that the number to the right of the decimal point specifies the input channel (see variable description). In the next example, shortened slot addresses (1.2, 1.3, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses and channel numbers. The slot address is expressed as a slot number to the left of the decimal point and a channel to the right. The slot address values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. The channel numbers run from 1 to 6. Thus, in order to read channel

5 of the 69423A card in slot 2 use: "402.5" or "2.5". If the slot address or channel numbers are not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc. — All even numbered variables are used to store data from the input cards.

pp0 — If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

p60-62 — Constants to convert from reference junction voltage to °C.

p50-53 — Constants to convert from reference junction °C to correction voltage of type E thermocouples.

p40-44 — Constants to convert from type E thermocouple voltage to °C.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram first reads the reference junction voltage and converts this to °C. This value in °C is then converted to a correction voltage for type E thermocouples. The specified channel of the addressed 69423A card is then read and converted to °C.

The conversion equation from °C of reference temperature to voltage correction factor is of the form:

$p50 + X\,(p51 + X\,(p52 + X\,p53)) = \mu V$

where, $p50 = a_0$, $p51 = a_1$, $p52 = a_2$, $p53 = a_3$.

The conversion equation from volts from the thermocouple junction to °C is of the form:

$p40 + X\,(p41 + X\,(p42 + X\,(p43 + p44))) = °C$

where, $p40 = a_0$, $p41 = a_1$, $p42 = a_2$, $p43 = a_3$, $p44 = a_4$.

The constants are taken from the Thermocouple Reference Tables Based on the IPTS — 68, U. S. Department of Commerce (March 1974). The constants chosen for this subprogram are for the following temperature range:

Temperature Range	Error Range
−20 to 500°C	−.18 to .12°C

For other ranges, change the p40-p44 constants to the values given in Table 3-4 for the applicable temperature range.

Table 3-4. Type E Thermocouples — Quartic Approximations to the data as a function of voltage in selected temperature ranges (°C). The expansion is of the form $T = a_0, a_1E, + a_2E^2 + a_3E^3 + a_4E^4$ where E is in microvolts and T is in degrees Celsius.

Temperature Range (°C)	a_0		a_1		a_2		a_3		a_4		Error Range (°C)
	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Exact-Approx.
-270 to 0	. . . . .	. .	2.8168878	-3	-8.5940057	-6	-1.4930918	-9	-8.7987588	-14	-9 to 6
-200 to 0	. . . . .	. .	1.5726646	-2	-1.2102152	-6	-1.9577799	-10	-1.6696298	-14	-.3 to .3
-200 to 800	. . . . .	. .	1.8432856	-2	-3.2311582	-7	6.9795810	-12	-5.1106852	-17	-8 to 7
- 20 to 500	. . . . .	. .	1.6970287	-2	-2.0830603	-7	4.6512717	-12	-4.1805785	-17	-.18 to .12
0 to 400	. . . . .	. .	1.7022525	-2	-2.2097240	-7	5.4809314	-12	-5.7669892	-17	-.05 to .04
0 to 1000	. . . . .	. .	1.6410783	-2	-1.3560189	-7	1.8600342	-12	-8.5537337	-18	-.9 to 1.4
400 to 1000	1.9669452	+1	1.4207735	-2	-5.1844510	-8	5.6361365	-13	-1.5646343	-18	-.03 to .03
600 to 800	2.5192188	+1	1.3909529	-2	-4.7201133	-8	5.5638718	-13	-1.7775228	-18	-.0005 to .0005
850 to 1000	-7.1102114	+2	5.6554599	-2	-9.7013068	-7	9.3938146	-12	-3.3333675	-17	-.001 to .001

SPECIAL CONSIDERATIONS

This subprogram returns temperature in °C. To convert to °F, use the following equation:

$$9/5 \times C + 32 = F$$

USED WITH

69423A card and type E thermocouples (Nickel-Chromium Alloy vs. Copper-Nickel Alloy)

PROGRAM LISTINGS

GPIO

```
0: "E Thermocouple":
1: "ET":esb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 58.6537565→p51;4.6720025e-2→p52;-1.4438022e-5→p53
4: 1.6970287e-2→p41;-2.0830603e-7→p42;4.6512717e-12→p43;-4.1805785e-17→p44
5: 1→p39
6: wtb "MP";170160+dtopp0
7: if (10frc(pp39)→p38)<7;sto +2
8: beep;prt "Address p";p39;"out of range";stp ;sto -1
9: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
10: wti 4,10000dtoint(pp39);rdi 4→p37
11: (otdp37-4096)*10^4352→p37
12: p60/(p61+ln(-p37))-p62→p37
13: p37(p51+p37(p52+p37p53))1e-6→p37
14: wtb "MP",170160+dtopp0,10000dtoint(pp39)+p38,170260+dtopp0
15: wti 4,10000dtoint(pp39);rdi 4→p36
16: ((otdp36→p36)-(p36<2047)4096)1e-5→p36
17: (p37+p36)1e6→p36
18: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
19: if (p39+2→p39)<=p0-1;sto -13
20: wtb "MP";170040;mdec
21: ret
*31034
```

Bytes used: 812

HP-IB

```
0: "E Thermocouple":
1: "ET":esb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 58.6537565→p51;4.6720025e-2→p52;-1.4438022e-5→p53
4: 1.6970287e-2→p41;-2.0830603e-7→p42;4.6512717e-12→p43;-4.1805785e-17→p44
5: 1→p39
6: wrt "MP.2","0",160+dtopp0,"T"
7: if (10frc(pp39)→p38)<7;sto +2
8: beep;prt "Address p";p39;"out of range";stp ;sto -1
9: wrt "MP.2",char(64+int(pp39)),7,"T","0",260+dtopp0,"T"
10: wrt "MP.1",char(64+int(pp39)),"X";red "MP",p37
11: (otdp37-4096)*10^4352→p37
12: p60/(p61+ln(-p37))-p62→p37
13: p37(p51+p37(p52+p37p53))1e-6→p37
14: wrt "MP.2","0",160+dtopp0,"T",char(int(pp39)+64),p38,"T"
15: wrt "MP.2","0",260+dtopp0,"T",char(64+int(pp39)),"X";red "MP",p36
16: ((otdp36→p36)-(p36<2047)4096)1e-5→p36
17: (p37+p36)1e6→p36
18: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
19: if (p39+2→p39)<=p0-1;sto -13
20: wrt "MP.1","040T"
21: ret
*25052
```

Bytes used: 864

File 37: R Thermocouple (RT)

Reads the temperature in °C from 69423A cards and type R thermocouples (platinum-13% rhodium alloy versus platinum). Each 69423A card has six input channels (see program examples.)

SYNTAX

cll 'R Thermocouple' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'RT' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot and Channel Address	None
D_n	p2, p4, etc.	Storage Locations for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'R Thermocouple' (401.2, A, 401.3, G, 402.5, B, 0)

or

cll 'R Thermocouple' (1.2, A, 1.3, G, 2.5, B)

or

cll 'RT' (1.2, A, 1.3, G, 2.5, B)

In the first example, the long descriptive label and the complete slot addresses (401.2, 401.3, etc.) are used, the data storage locations are specified, and the unit number is included. Note that the number to the right of the decimal point specifies the input channel (see variable description). In the next example, shortened slot addresses (1.2, 1.3, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc.

All odd numbered variables are used for slot addresses and channel numbers. The slot address is expressed as a slot number to the left of the decimal point and a channel to the right. The slot address values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. The channel numbers run from 1 to 6. Thus, in order to read channel 5 of the 69423A card in slot 2 use: "402.5" or "2.5". If the slot address or channel numbers are not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even numbered variables are used to store data from the input cards.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.
p60-62	Constants to convert from reference junction voltage to °C.
p50-53	Constants to convert from reference junction °C to correction voltage of type R thermocouples.
p40-44	Constants to convert from type R thermocouple voltage to °C.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram first reads the reference junction voltage and converts this to °C. This value in °C is then converted to a correction voltage for type R thermocouples. The specified channel of the addressed 69423A card is then read and converted to °C.

The conversion equation from °C of reference temperature to voltage correction factor is of the form:

$p50 + X\,(p51 + X\,(p52 + X\,p53)) = \mu V$

where, $p50 = a_0$, $p51 = a_1$, $p52 = a_2$, $p53 = a_3$.

The conversion equation from volts from the thermocouple junction to °C is of the form:

$p40 + X\,(p41 + X\,(p42 + X\,(p43 + p44))) = °C$

where, $p40 = a_0$, $p41 = a_1$, $p42 = a_2$, $p43 = a_3$, $p44 = a_4$.

The constants are taken from the Thermocouple Reference Tables Based on the IPTS — 68, U. S. Department of Commerce (March 1974). The constants chosen for this subprogram are for the following temperature range:

Temperature Range	Error Range
400-1100°C	— .04 to .04°C

For other ranges, change the p40-p44 constants to the values given in Table 3-5 for the applicable temperature range.

Table 3-5. Type R Thermocouples — Quartic Approximations to the data as a function of voltage in selected temperature ranges (°C). The expansion is of the form $T = a_0, a_1E, + a_2E^2 + a_3E^3 + a_4E^4$ where E is in microvolts and T is in degrees Celsius.

Temperature Range (°C)	a_0		a_1		a_2		a_3		a_4		Error Range (°C)
	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Exact-Approx.
- 50 to 900	. . . . .	. .	1.6251434	-1	-2.0454379	-5	2.5404935	-9	-1.1767904	-13	-13 to 3
0 to 1100	. . . . .	. .	1.5239494	-1	-1.3755675	-5	1.2610922	-9	-4.4281251	-14	- 4 to 7
0 to 1400	. . . . .	. .	1.4441607	-1	-9.5014952	-6	6.2073358	-10	-1.5622497	-14	- 6 to 10
0 to 1650	. . . . .	. .	1.3944190	-1	-7.4485484	-6	3.8266182	-10	-7.4517277	-15	- 7 to 13
0 to 1768	. . . . .	. .	1.3752883	-1	-6.7651171	-6	3.1420473	-10	-5.4254872	-15	- 7 to 14
400 to 1100	4.5509556	+1	1.1284875	-1	-2.8603978	-6	8.5173702	-11	-1.1440038	-15	-.04 to .04
400 to 1400	4.9160016	+1	1.1054589	-1	-2.3559046	-6	3.9276248	-11	3.3369324	-16	-.08 to .09
400 to 1650	4.8343651	+1	1.1098270	-1	-2.4353890	-6	4.5164488	-11	1.8172612	-16	-.10 to .12
1050 to 1400	-4.1134459	+0	1.2738464	-1	-4.3132296	-6	1.3863582	-10	-1.5283798	-15	-.002 to .002
1050 to 1650	3.7487318	+1	1.1519304	-1	-2.9827002	-6	7.4538667	-11	-3.7809957	-16	-.011 to .011
1400 to 1550	8.0559850	+1	1.0442877	-1	-1.9827500	-6	3.3603790	-11	2.4513433	-16	-.0005 to .0005
1400 to 1650	1.4180146	+2	9.0181346	-2	-7.4068329	-7	-1.4487255	-11	9.4290495	-16	-.0005 to .0005
1400 to 1768	3.1759093	+3	-5.8922431	-1	5.6190639	-5	-2.1303241	-9	3.0369250	-14	-.11 to .08
1666 to 1768	1.2883437	+4	-2.6747958	+0	2.2334214	-4	-8.0565860	-9	1.0882779	-13	-.0007 to .0007

SPECIAL CONSIDERATIONS

This subprogram returns temperature in °C. To convert to °F, use the following equation:

$$9/5 \times C + 32 = F$$

USED WITH

69423A card and type R thermocouples (platinum-13% rhodium alloy vs. platinum)

PROGRAM LISTINGS

GPIO

```
0: "R Thermocouple":
1: "RT":ssb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 5.2891411→p51;1.3844426e-2→p52;-2.0889531e-5→p53
4: 4.5509556→p40;.11284875→p41;-2.8603978e-6→p42;8.5173702e-11→p43
5: -1.1440038e-15→p44
6: 1→p39
7: wtb "MP",170160+dtopp0
8: if (10frc(pp39)→p38)<7;sto +2
9: beep;prt "Address p",p39,"out of range";stp ;sto -1
10: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
11: wti 4,10000dtoint(pp39);rdi 4→p37
12: (otdp37-4096)*10/4352→p37
13: p60/(p61+ln(-p37)/-p62→p37
14: p37(p51+p37(p52+p37p53))1e-6→p37
15: wtb "MP",170160+dtopp0,10000dtoint(pp39)+p38,170260+dtopp0
16: wti 4,10000dtoint(pp39);rdi 4→p36
17: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
18: (p37+p36)1e6→p36
19: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
20: if (p39+2→p39)<=p0-1;sto -13
21: wtb "MP",170040;mdec
22: ret
*21837
```

Bytes used: 826

HP-IB

```
0: "R Thermocouple":
1: "RT":ssb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 5.2891411→p51;1.3844426e-2→p52;-2.0889531e-5→p53
4: 4.5509556→p40;.11284875→p41;-2.8603978e-6→p42;8.5173702e-11→p43
5: -1.1440038e-15→p44
6: 1→p39
7: wrt "MP.2","0",160+dtopp0,"T"
8: if (10frc(pp39)→p38)<7;sto +2
9: beep;prt "Address p",p39,"out of range";stp ;sto -1
10: wrt "MP.2",char(64+int(pp39)),7,"T","0",260+dtopp0,"T"
11: wrt "MP.1",char(64+int(pp39)),"X";red "MP",p37
12: (otdp37-4096)*10/4352→p37
13: p60/(p61+ln(-p37)/-p62→p37
14: p37(p51+p37(p52+p37p53))1e-6→p37
15: wrt "MP.2","0",160+dtopp0,"T",char(int(pp39)+64),p38,"T"
16: wrt "MP.2","0",260+dtopp0,"T",char(64+int(pp39)),"X";red "MP",p36
17: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
18: (p37+p36)1e6→p36
19: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
20: if (p39+2→p39)<=p0-1;sto -13
21: wrt "MP.1","040T"
22: ret
*14758
```

Bytes used: 878

File 38: S Thermocouple (ST)

Reads the temperature in °C from 69423A cards and type S thermocouples (platinum - 10% rhodium alloy vs. platinum). Each 69423A card has six input channels (see program examples.)

SYNTAX

cll 'S Thermocouple' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'ST' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot and Channel Address	None
D_n	p2, p4, etc.	Storage Locations for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'S Thermocouple' (401.2, A, 401.3, G, 402.5, B, 0)

or

cll 'S Thermocouple' (1.2, A, 1.3, G, 2.5, B)

or

cll 'ST' (1.2, A, 1.3, G, 2.5, B)

In the first example, the long descriptive label and the complete slot addresses (401.2, 401.3, etc.) are used, the data storage locations are specified, and the unit number is included. Note that the number to the right of the decimal point specifies the input channel (see variable description). In the next example, shortened slot addresses (1.2, 1.3, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses and channel numbers. The slot address is expressed as a slot number to the left of the decimal point and a channel to the right. The slot

address values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. The channel numbers rum from 1 to 6. Thus, in order to read channel 5 of the 69423A card in slot 2 use: "402.5" or "2.5". If the slot address or channel numbers are not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even numbered variables are used to store data from the input cards.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.
p60-62	Constants to convert from reference junction voltage to °C.
p50-53	Constants to convert from reference junction °C to correction voltage of type S thermocouples.
p40-44	Constants to convert from type S thermocouple voltage to °C.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram first reads the reference junction voltage and converts this to °C. This value in °C is then converted to a correction voltage for type S thermocouples. The specified channel of the addressed 69423A card is then read and converted to °C.

The conversion equation from °C of reference temperature to voltage correction factor is of the form:
$p50 + X\ (p51 + X\ (p52 + X\ p53)) = \mu V$
where, $p50 = a_0$, $p51 = a_1$, $p52 = a_2$, $p53 = a_3$.

The conversion equation from volts from the thermocouple junction to °C is of the form:
$p40 + X\ (p41 + X\ (p42 + X\ (p43 + p44))) = °C$
where, $p40 = a_0$, $p41 = a_1$, $p42 = a_2$, $p43 = a_3$, $p44 = a_4$.

The constants are taken from the Thermocouple Reference Tables Based on the IPTS — 68, U. S. Department of Commerce (March 1974). The constants chosen for this subprogram are for the following temperature range:

Temperature Range	Error Range
400 to 1400°C	— .08 to .08°C

For other ranges, change the p40-p44 constants to the values given in Table 3-6 for the applicable temperature range.

Table 3-6. Type S Thermocouples — Quartic Approximations to the data as a function of voltage in selected temperature ranges (°C). The expansion is of the form $T = a_0, a_1E, + a_2E^2 + a_3E^3 + a_4E^4$ where E is in microvolts and T is in degrees Celsius.

Temperature Range (°C)	a_0		a_1		a_2		a_3		a_4		Error Range (°C)
	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Exact-Approx.
- 50 to 900	. . . . .	. .	1.6414048	-1	-2.0241757	-5	2.7849728	-9	-1.4172102	-13	-11 to 3
0 to 1100	. . . . .	. .	1.5445376	-1	-1.3349067	-5	1.3626587	-9	-5.3270847	-14	-3 to 6
0 to 1400	. . . . .	. .	1.4713897	-1	-9.0783455	-6	6.5660913	-10	-1.8499175	-14	-5 to 9
0 to 1650	. . . . .	. .	1.4260554	-1	-7.0073775	-6	3.8981279	-10	-8.3047780	-15	-6 to 11
0 to 1768	. . . . .	. .	1.4087955	-1	-6.3195007	-6	3.1267454	-10	-5.7422562	-15	-6 to 12
400 to 1100	4.1137317	+1	1.1599785	-1	-1.8642979	-6	1.2643267	-11	8.4828836	-16	-.05 to .07
400 to 1400	4.4507790	+1	1.1373998	-1	-1.3349811	-6	-3.9224680	-11	2.6563405	-15	-.08 to .08
400 to 1650	4.1670535	+1	1.1543356	-1	-1.6782780	-6	-1.0845801	-11	1.8379726	-15	-.2 to .2
1050 to 1400	-3.0938374	+1	1.4106560	-1	-4.9794442	-6	1.7334256	-10	-1.9262160	-15	-.003 to .003
1050 to 1650	1.2226507	+1	1.2706383	-1	-3.2873314	-6	8.3038098	-11	-1.3019379	-16	-.010 to .010
1400 to 1550	1.3866867	+2	9.3486676	-2	4.8592708	-8	-6.3885209	-11	2.2896541	-15	-.0005 to .0005
1400 to 1650	1.3923740	+2	9.3267401	-2	7.7266682	-8	-6.5458208	-11	2.3208160	-15	-.0005 to .0005
1400 to 1768	4.5133695	+3	-1.0046437	+0	1.0322002	-4	-4.3637046	-9	6.9361610	-14	-.13 to .10
1666 to 1768	2.3131446	+4	-5.4122671	+0	4.9347196	-4	-1.9681943	-8	2.9430179	-13	-.0005 to .0005

SPECIAL CONSIDERATIONS

This subprogram returns temperature in °C. To convert to °F, use the following equation:

$$9/5 \times C + 32 = F$$

USED WITH

69423A card and type S thermocouples (platinum - 10% rhodium alloy vs. platinum)

PROGRAM LISTINGS

GPIO

```
0: "S Thermocouple":
1: "ST":gsb "Parameter Check"
2: 3923.7225→p60;13.153→p61;273.2→p62
3: 5.3994446→p51;1.2467754e-2→p52;-1.9934168e-5→p53
4: 44.50779→p40;.11373998→p41;-1.3349811e-6→p42;-3.922468e-11→p43
5: 2.6563405e-15→p44
6: 1→p39
7: wtb "MP",170160+dtopp0
8: if (10frc(pp39)→p38)<7;gto +2
9: beep;prt "Address p",p39,"out of range";stp ;gto -1
10: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
11: wti 4,10000dtoint(pp39);rdi 4→p37
12: (otdp37-4096)*10/4352→p37
13: p60-(p61+ln(-p37))-p62→p37
14: p37(p51+p37(p52+p37p53))1e-6→p37
15: wtb "MP",170160+dtopp0,10000dtoint(pp39)+p38,170260+dtopp0
16: wti 4,10000dtoint(pp39);rdi 4→p36
17: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
18: (p37+p36)1e6→p36
19: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
20: if (p39+2→p39)<=p0-1;gto -13
21: wtb "MP",170040;mdec
22: ret
-13815
```

Bytes used: 826

File 38: S Thermocouple (ST)

HP-IB

```
0: "S Thermocouple":
1: "ST":ssb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 5.3994446→p51;1.2467754e-2→p52;-1.9934168e-5→p53
4: 44.50779→p40;.11373996→p41;-1.3349811e-6→p42;-3.922468e-11→p43
5: 2.6563405e-15→p44
6: 1→p39
7: wrt "MP.2","O",160+dtopp0,"T"
8: if (10frc(pp39)→p38)<7;sto +2
9: beep;prt "Address p",p39,"out of range";stp ;sto -1
10: wrt "MP.2",char(64+int(pp39)),7,"T","O",260+dtopp0,"T"
11: wrt "MP.1",char(64+int(pp39)),"X";red "MP",p37
12: (otdp37-4096)*10^4352→p37
13: p60/(p61+ln(-p37))-p62→p37
14: p37(p51+p37(p52+p37p53))1e-6→p37
15: wrt "MP.2","O",160+dtopp0,"T",char(int(pp39)+64),p38,"T"
16: wrt "MP.2","O",260+dtopp0,"T",char(64+int(pp39)),"X";red "MP",p36
17: ((otdp36→p36)-(p36)2047)4096)1e-5→p36
18: (p37+p36)1e6→p36
19: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
20: if (p39+2→p39)<=p0-1;sto -13
21: wrt "MP.1","040T"
22: ret
*6736
```

Bytes used: 878

Reads the temperature in °C of the reference on the 69423A card.

SYNTAX

cll 'Thermocouple Reference' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'TR' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot Address	None
D_n	p2, p4, etc.	Storage Location for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'Thermocouple Reference' (401, A, 407, G, 402, B, 0)

or

cll 'Thermocouple Reference' (1, A, 7, G, 2, B)

or

cll 'TR' (401, A, 407, G, 402, B)

In the first example, the long descriptive label and the complete slot addresses (401, 407, etc.) are used, the data storage locations are specified, and the unit number is included. In the next example, shortened slot addresses (1, 7, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

File 39: Thermocouple Reference (TR)

VARIABLE DESCRIPTION

p1, p3, etc.	All odd numbered variables are used for slot addresses. Their values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. If this parameter is not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.
p2, p4, etc.	All even number variables are used to store data from input cards.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0).

This subprogram reads the reference on the specified 69423A and returns the temperature in °C. The reference junction is channel 7 on the 69423A.

SPECIAL CONSIDERATIONS

The temperature is returned in °C. To convert to °C use the following equation;

$$9/5 \times C + 32 = F$$

USED WITH

69423A Low Level Analog Input Card.

PROGRAM LISTINGS

GPIO

```
0: "Thermocouple Reference":
1: "TR":gsb "Parameter Check"
2: 3923.7225+pp0;13.108+p61;273.2+p62
3: 1→p39
4: wtb "MP",170160+dtopp0
5: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
6: wti 4,10000dtoint(pp39);rdi 4→p37
7: (otdp37-4096)*10/4352→p37
8: p60/(p61+ln(-p37))-p62→p(p39+1)
9: if (p39+2→p39)<=p0-1;gto -5
10: wtb "MP",170040;mdec
11: ret
*19655
```

Bytes used: 350

File 39: Thermocouple Reference (TR)

HP-IB

```
0: "Thermocouple Reference":
1: "TR":esb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: 1→p39
4: wrt "MP.2","O",160+dtopp0,"T"
5: wrt "MP.2",char(64+int(pp39)),7,"T","O",260+dtopp0,"T"
6: wrt "MP.1",char(64+int(pp39)),"X";red "MP",p37
7: (otdp37-4096)*10/4352→p37
8: p60/(p61+ln(-p37))-p62→p(p39+1)
9: if (p39+2→p39)<=p0-1;gto -5
10: wrt "MP.1","040T"
11: ret
*9872
```

Bytes used: 376

File 40: B Thermocouple (BT)

Reads the temperature in °C from 69423A cards and type B thermocouples (platinum - 30% rhodium alloy vs. platinum - 6% rhodium alloy). Each 69423A card has six input channels (see program examples.)

SYNTAX

cll 'B Thermocouple' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'BT' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot and Channel Address	None
D_n	p2, p4, etc.	Storage Locations for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'B Thermocouple' (401.2, A, 401.3, G, 402.5, B, 0)

or

cll 'B Thermocouple' (1.2, A, 1.3, G, 2.5, B)

or

cll 'BT' (1.2, A, 1.3, G, 2.5, B)

In the first example, the long descriptive label and the complete slot addresses (401.2, 401.3, etc.) are used, the data storage locations are specified, and the unit number is included. Note that the number to the right of the decimal point specifies the input channel (see variable description). In the next example, shortened slot addresses (1.2, 1.3, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses and channel numbers. The slot address is expressed as a slot number to

the left of the decimal point and a channel to the right. The slot address values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. The channel numbers run from 1 to 6. Thus, in order to read channel 5 of the 69423A card in slot 2 use: "402.5" or "2.5". If the slot address or channel numbers are not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.

p2, p4, etc.	All even numbered variables are used to store data from the input cards.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.
p60-62	Constants to convert from reference junction voltage to °C.
p50-53	Constants to convert from reference junction °C to correction voltage of type B thermocouples.
p40-44	Constants to convert from type B thermocouple voltage to °C.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram first reads the reference junction voltage and converts this to °C. This value in °C is then converted to a correction voltage for type B thermocouples. The specified channel of the addressed 69423A card is then read and converted to °C.

The conversion equation from °C of reference temperature to voltage correction factor is of the form:
$p50 + X\ (p51 + X\ (p52 + X\ p53)) = \mu V$
where, $p50 = a_0$, $p51 = a_1$, $p52 = a_2$, $p53 = a_3$.

The conversion equation from volts from the thermocouple junction to °C is of the form:
$p40 + X\ (p41 + X\ (p42 + X\ (p43 + p44))) = °C$
where, $p40 = a_0$, $p41 = a_1$, $p42 = a_2$, $p43 = a_3$, $p44 = a_4$.

The constants are taken from the Thermocouple Reference Tables Based on the IPTS — 68, U. S. Department of Commerce (March 1974). The constants chosen for this subprogram are for the following temperature range:

Temperature Range	Error Range
400 to 1100°C	−.09 to 1.0°C

For other ranges, change the p40-p44 constants to the values given in Table 3-7 for the applicable temperature range.

Table 3-7. Type B Thermocouples — Quartic Approximations to the data as a function of voltage in selected temperature ranges (°C). The expansion is of the form $T = a_0, a_1E, + a_2E^2 + a_3E^3 + a_4E^4$ where E is in microvolts and T is in degrees Celsius.

Temperature Range (°C)	a_0		a_1		a_2		a_3		a_4		Error Range (°C)
	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Exact-Approx.
0 - 900	. . . . .		8.9244743	-1	-5.7447033	-4	1.8053618	-7	-1.9719121	-11	-30 to 75
0 - 1100	. . . . .		7.2874066	-1	-3.1771931	-4	6.8254996	-8	-5.1002233	-12	-35 to 90
0 - 1400	. . . . .		5.7822214	-1	-1.6039309	-4	2.2187592	-8	-1.0678514	-12	-45 to 110
0 - 1650	. . . . .		4.9929130	-1	-1.0349686	-4	1.0792281	-8	-3.9111456	-13	-50 to 120
0 - 1820	. . . . .		4.6255054	-1	-8.2176262	-5	7.3717195	-9	-2.2913665	-13	-50 to 130
400 - 1100	1.8946288	+2	3.0966136	-1	-5.8100680	-5	8.2483967	-9	-4.7591774	-13	-.09 to 1.0
400 - 1400	2.0949015	+2	2.7222162	-1	-3.6930932	-5	3.6830239	-9	-1.4483702	-13	-3 to 3
400 - 1650	2.2354664	+2	2.4988761	-1	-2.7160312	-5	2.1299660	-9	-6.4220755	-14	-5 to 5
1050 - 1400	3.2188156	+2	1.8282378	-1	-1.1561743	-5	6.4320083	-10	-1.4544375	-14	-.003 to .003
1050 - 1650	3.4418084	+2	1.7031473	-1	-8.9696912	-6	4.0789445	-10	-6.6410259	-15	-.025 to .020
1400 - 1550	3.7140306	+2	1.5828913	-1	-7.0050689	-6	2.6714849	-10	-2.9082072	-15	-.001 to .001
1400 - 1650	3.9253848	+2	1.4979551	-1	-5.7276293	-6	1.8192801	-10	-7.8042686	-16	-.001 to .001

SPECIAL CONSIDERATIONS

This subprogram returns temperature in °C. To convert to °F, use the following equation:

$$9/5 \times C + 32 = F$$

USED WITH

69423A card and type B thermocouples (platinum - 30% rhodium alloy vs. platinum - 6% rhodium alloy.

PROGRAM LISTINGS

GPIO

```
0: "B Thermocouple":
1: "BT":asb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: -.24673839→p51;5.9050303e-3→p52;-1.226718e-6→p53
4: 1.8946288e2→p40;.30966136→p41;-5.810068e-5→p42
5: 8.2483967e-9→p43;-4.7591774e-13→p44
6: 1→p39
7: wtb "MP",170160+dtopp0
8: if (10frc(pp39)→p39)>7;ato +2
9: beep;prt "Address p",p39,"out of range";stp ;ato -1
10: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
11: wti 4,10000dtoint(pp39);rdi 4→p37
12: (otdp37-4096)*10 4052→p37
13: p60/(p61+ln(-p37))-p62→p37
14: p37(p51+p37(p52+p37p53))1e-6→p37
15: wtb "MP",170160+dtopp0,10000dtoint(pp39)+p38,170260+dtopp0
16: wti 4,10000dtoint(pp39);rdi 4→p36
17: ((otdp36→p36)-(p36 2047)4096)1e-5→p36
18: (p37+p36)1e6→p36
19: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
20: if (p39+2→p39)<=p0-1;ato -13
21: wtb "MP",170040;mdec
22: ret
→31134
```

Bytes used: 830

HP-IB

```
0: "B Thermocouple":
1: "BT":ssb "Parameter Check"
2: 3923.7225→p60;13.158→p61;273.2→p62
3: -.24673839→p51;5.9050303e-3→p52;-1.226718e-6→p53
4: 1.8946288e2→p40;.30966136→p41;-5.810068e-5→p42
5: 8.2483967e-9→p43;-4.7591774e-13→p44
6: 1→p39
7: wrt "MP.2","0",160+dtopp0,"T"
8: if (10frc(pp39)→p38)<7;gto +2
9: beep;prt "Address p",p39,"out of range";stp ;gto -1
10: wrt "MP.2",char(64+int(pp39)),7,"T","0",260+dtopp0,"T"
11: wrt "MP.1",char(64+int(pp39)),"X";red "MP",p37
12: (otdp37-4096)*10^4352→p37
13: p60/(p61+ln(-p37))-p62→p37
14: p37(p51+p37(p52+p37p53))1e-6→p37
15: wrt "MP.2","0",160+dtopp0,"T",char(int(pp39)+64),p38,"T"
16: wrt "MP.2","0",260+dtopp0,"T",char(64+int(pp39)),"X";red "MP",p36
17: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
18: (p37+p36)1e6→p36
19: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
20: if (p39+2→p39)<=p0-1;gto -13
21: wrt "MP.1","040T"
22: ret
*24055
```

Bytes used: 882

File 41: K Thermocouple (KT)

Reads the temperature in °C from 69423A cards and type K thermocouples (nickel - chromium alloy vs. nickel - aluminum alloy). Each 69423A card has six input channels (see program examples.)

SYNTAX

cll 'K Thermocouple' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

or

cll 'KT' (S_0, D_0, S_1, D_1, - - - - S_n, D_n, [U])

PARAMETERS

Parameter List	Variable	Description	Default Value
S_n	p1, p3, etc.	Card Slot and Channel Address	None
D_n	p2, p4, etc.	Storage Locations for Data	None
U	pp0	Unit Number	Unit 0
	p0	Number of Parameters Passed	None

PROGRAM EXAMPLES

cll 'K Thermocouple' (401.2, A, 401.3, G, 402.5, B, 0)

or

cll 'K Thermocouple' (1.2, A, 1.3, G, 2.5, B)

or

cll 'KT' (1.2, A, 1.3, G, 2.5, B)

In the first example, the long descriptive label and the complete slot addresses (401.2, 401.3, etc.) are used, the data storage locations are specified, and the unit number is included. Note that the number to the right of the decimal point specifies the input channel (see variable description). In the next example, shortened slot addresses (1.2, 1.3, etc.) are used, and the optional unit number is omitted. Finally, a short two letter abbreviation is used for the "cll" label, and the optional unit number is omitted. The subprogram will automatically assign a value of zero to pp0 when the unit number is omitted.

VARIABLE DESCRIPTION

p1, p3, etc. — All odd numbered variables are used for slot addresses and channel numbers. The slot address is expressed as a slot number to

	the left of the decimal point and a channel to the right. The slot address values must be from 400 to 414 or from 0 (zero) to 14, where 1 is understood to mean 401, 2 is used for 402, etc. The channel numbers run from 1 to 6. Thus, in order to read channel 5 of the 69423A card in slot 2 use: "402.5" or "2.5". If the slot address or channel numbers are not within range, the program will stop and permit entry of a new value. Slot addresses must be in decimal.
p2, p4, etc.	All even numbered variables are used to store data from the input cards.
pp0	If the unit number is greater than 15, the program will stop and permit entry of a new value. Unit number must be in decimal.
p60-62	Constants to convert from reference junction voltage to °C.
p50-53	Constants to convert from reference junction °C to correction voltage of type K thermocouples.
p40-44	Constants to convert from type K thermocouple voltage to °C.

PROGRAM DESCRIPTION

The subprogram starts with a "gsb" to 'Parameter Check'. The 'Parameter Check' subprogram checks each parameter to ensure that they are within limits. Parameters out of limits will cause the program to stop and wait for a new value.

Upon returning from 'Parameter Check', this subprogram sends a control word to the unit designated by pp0. The unit number, however, is first converted to octal (dto pp0). The subprogram first reads the reference junction voltage and converts this to °C. This value in °C is then converted to a correction voltage for type T thermocouples. The specified channel of the addressed 69423A card is then read and converted to °C.

The conversion equation from °C of reference temperature to voltage correction factor is of the form:
$p50 + X (p51 + X (p52 + X p53)) = \mu V$
where, $p50 = a_0$, $p51 = a_1$, $p52 = a_2$, $p53 = a_3$.

The conversion equation from volts from the thermocouple junction to °C is of the form:
$p40 + X (p41 + X (p42 + X (p43 + p44))) = °C$
where, $p40 = a_0$, $p41 = a_1$, $p42 = a_2$, $p43 = a_3$, $p44 = a_4$.

The constants are taken from the Thermocouple Reference Tables Based on the IPTS — 68, U. S. Department of Commerce (March 1974). The constants chosen for this subprogram are for the following temperature range:

Temperature Range	Error Range
−20 to 500°C	−1.2 to 0.6°C

For other ranges, change the p40-p44 constants to the values given in Table 3-8 for the applicable temperature range.

Table 3-8. Type K Thermocouples — Quartic Approximations to the data as a function of voltage in selected temperature ranges (°C). The expansion is of the form $T = a_0, a_1E, + a_2E^2 + a_3E^3 + a_4E^4$ where E is in microvolts and T is in degrees Celsius.

Temperature Range (°C)	a_0		a_1		a_2		a_3		a_4		Error Range (°C)
	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Argument	Exp.	Exact-Approx.
-270 to 0	.....	..	1.2329875	-2	-1.4434305	-5	-4.2824995	- 9	-4.2028679	-13	-11 to 8
-200 to 0	.....	..	2.3783697	-2	-2.4382217	-6	-6.8203073	-10	-9.4854031	-14	-.5 to .5
-200 to 800	.....	..	2.8346886	-2	-5.8008526	-7	2.5720615	-11	-3.6813679	-16	-8 to 10
- 20 to 500	.....	..	2.4363851	-2	5.6206931	-8	-3.8825620	-12	3.9120208	-17	-1.2 to .6
0 to 400	.....	..	2.4383248	-2	9.7830251	-9	3.6276965	-12	-2.5756438	-16	-.5 to .6
0 to 1370	.....	..	2.5132785	-2	-6.0883423	-8	5.5358209	-13	9.3720918	-18	-2.4 to 1.2
400 to 1000	-2.4707112	+1	2.9465633	-2	-3.1332620	-7	6.5075717	-12	-3.9663834	-17	-.02 to .02
400 to 1370	6.2300671	+0	2.4955374	-2	-7.8788333	-8	1.3269743	-12	1.5580541	-18	-.3 to .3
600 to 800	-3.9480992	+1	3.1425797	-2	-4.0905633	-7	8.5482602	-12	-5.5696636	-17	-.001 to .001
850 to 1000	-3.1617495	+0	2.7115517	-2	-2.1941995	-7	4.8782826	-12	-2.9316611	-17	-.0012 to .0012
1050 to 1150	2.3615582	+2	1.1066277	-3	8.2516607	-7	-1.3558849	-11	9.1638500	-17	-.001 to .001

SPECIAL CONSIDERATIONS

This subprogram returns temperature in °C. To convert to °F, use the following equation:

$$9/5 \times C + 32 = F$$

USED WITH

69423A card and type K thermocouples (nickel - chromium alloy vs. nickel - aluminum alloy).

PROGRAM LISTINGS

GPIO

```
0: "K Thermocouple":
1: "KT":gsb "Parameter Check"
2: [illegible];13.158→p61;273.2→p62
3: [illegible]→p51;2.4548362e-2→p52; 9.0918433e-6→p53
4: [illegible]e-2→p41;5.6206931e-8→p42
5: [illegible]e-12→p43;3.9120208e-17→p44
6: 1→p39
7: wtb "MP",170160+dtopp0
8: if (10frc(pp39)→p38)<7;gto +2
9: beep;prt "Address p",p39,"out of range";stp ;gto -1
10: wtb "MP",10000dtoint(pp39)+7,170260+dtopp0
11: [illegible]dtoint(pp39);rdi 4→p37
12: [illegible]→10 4352→p37
13: [illegible]→p37 [illegible]→p62→p37
14: [illegible]p52→p37p53)1e-6→p37
15: [illegible]+dtopp0,10000dtoint(pp39)+p38,170260+dtopp0
16: wti 4,10000dtoint(pp39);rdi 4→p36
17: ((otdp[illegible] 2047)4096)1e 5→p36
18: [illegible]
19: [illegible]p42+p36(p43+p36p44)[illegible]p39+1
20: if [illegible];gto -13
21: [illegible]
22: [illegible]
→396
```

Bytes used: 814

HP-IB

```
0: "K Thermocouple":
1: "KT":ssb "Parameter Check"
2: 3423.7225→p60;13.158→p61;273.2→p62
3: 39.448872→p51;2.4542362e-2→p52;-9.0918433e-6→p53
4: 2.4363851e-2→p41;5.6206931e-8→p42
5: -3.8825b2e-12→p43;3.9120208e-17→p44
6: 1→p39
7: wrt "MP.2","O",160+dtopp0,"T"
8: if (10frc(pp39)→p38)<7;sto +2
9: beep;prt "Address p",p39,"out of range";stp ;sto -1
10: wrt "MP.2",char(64+int(pp39)),7,"T","O",260+dtopp0,"1"
11: wrt "MP.1",char(64+int(pp39)),"X";red "MP",p37
12: (otdp37-4096)*10/4352→p37
13: p60/(p61+ln(-p37))-p62→p37
14: p37(p51+p37(p52+p37p53))1e-6→p37
15: wrt "MP.2","O",160+dtopp0,"T",char(int(pp39)+64),p38,"T"
16: wrt "MP.2","O",260+dtopp0,"T",char(64+int(pp39)),"X";red "MP",p36
17: ((otdp36→p36)-(p36>2047)4096)1e-5→p36
18: (p37+p36)1e6→p36
19: p40+p36(p41+p36(p42+p36(p43+p36p44)))→p(p39+1)
20: if (p39+2→p39)<=p0-1;sto -13
21: wrt "MP.1","040T"
22: ret
+2317
```

Bytes used: 866

APPENDIX A

CHANGING THE INTERFACE SELECT CODES

As shipped from the factory, the interfaces for the 6940B Multiprogrammer have the following select codes:

Interface	Select Code
GPIO (Model 98032A-Opt.040)	9
HP-IB (Model 98034A)	7 (and bus address 23)

If a conflict arises whereby the select codes must be changed, the following procedure should be followed.

First the rotary switch on the interface is changed to the new desired select code. The second step is a software modification. Line 1 in the library subprogram entitled "Parameter Check" (file 2) associates "MP" with the proper select code (and bus address if (HP-IB). As stated above, the normal select codes are 9 for GPIO and 723 for HP-IB. This number must be changed to correspond with the new select code. For example, if a new select code of 6 is desired for the HP-IB, the new line 1 of "Parameter Check" on trk 1 becomes:

```
"PC": if flg 10#1; dev "MP", 623, "HP-IB", 6;sfg 10
```

To change the GPIO select code to 8, change line 1 of "Parameter Check" on trk 0 to:

```
"PC": if flg 10#1; mdec; dev "MP", 8;wti 0,8;sfg 10
```

The corrected "Parameter Check" should be stored back onto the Software Library Cartridge in file 2 on the proper trk (0 for GPIO, 1 for HP-IB).

APPENDIX B

ADDING CALL SUBPROGRAMS

Programs written using call subprograms not contained in the 14556A Software Library will not run until the call subprograms have been added to the program prepared by the Auto Loader. The subprograms can be added using one of the following methods:

1. Enter the required call subprogram as part of the original user application program.

OR

2. Load the required call subprograms into memory (at the end of the existing program lines) from data cartridge. See example below.

Example: To load a call subprogram from file 50, track 0 to the end of a program already stored in 9825A memory, perform the following steps.

a. Execute the following statement: fetch 9999

b. Press the display arrow, ↑, to view the last line of the program.

c. Note the line number of the last line of program (e.g., 72).

d. Insert the data cartridge containing the call subprogram that you want to load.

e. Execute the following statement: ldf 50, —

Insert a number equal to one more than that noted in step c (e.g., 1 + 72 = 73).

OR

3. Update the 14556A Software Library Directory prior to running the Auto-Loader. The Auto Loader will then automatically load the required call subprogram(s). The directory is part of the Auto Loader program (File No. 1) and contains the file number and call labels of all subprograms contained in the library. Directory updates can be accomplished quite easily. Basically, the string variable C$, that contains the directory is updated and then, if necessary, the variable S in line 0 is updated. To update the directory, proceed as follows:

a. Insert 14556A cartridge.

1. Select track 0 for GPIO or track 1 for HP-IB interface.
2. Load the Auto Loader from file 1 of the library.
3. Execute the following statement: fetch 9999

4. Press the display arrow, ↑, to view the last line of the program.

b. Update String Variable C$.

1. Delete last statement: gto 1.
2. Concatenate the update information with the existing string C$. Each entry must start with a two digit number indicating file number, and end with the label for the call subprogram.
3. Add the following statement: gto 1.

c. Update Variable S.

1. Note the line number of your last line of Auto Loader.
2. Fetch line 0.
3. Change variable S to a number that is one greater than the last line number of the Auto Loader.
4. Change the print statement of line 35 to: del 0, ——— *

last line number of Auto Loader

d. Record the updated Auto Loader in file 1 by executing: rcf 1

e. Record new call subprograms on appropriate track and file.

INTERIM USERS GUIDE FOR THE HP 6940B WITH 9835 AND 9845 DESKTOP COMPUTERS

This guide may be used in conjunction with the 9825A Users Guide (59500-90005) as an aid to 9835 and 9845 programmers.

50-1827

OUTPUT

A) 16-bit duplex I/O (98032A opt 040, 340, or 440)

1. Date to be sent to the 6940B is specified as a decimal number. The most significant bit (MSB) is the sign bit.
2. Control words are given on the attached Decimal Control Word Chart (page 4).
3. The following commands may be used:

OUTPUT 9 WHS USING "#,W"; –4064
The data sent, –4064, is a control word in decimal.

WRITE BIN 9 ; –4064
This command will work with the 9835A/B and the 9845B/T.

OUTPUT 9 type **NO FORMAT**; list
This is for block transfers. The variables in the list must be dimensioned as intergers:
type = **WINT** - interrupt
WFHS - fast handshake
WDMA - DMA
WHS - handshake

B) HP-IB

1. To send a string over the HP-IB to the Multiprogrammer the **K** image can be used to send the entire string, or **nA** may be used to send n characters.
2. The following commands may be used:

```
10 IMAGE #, K
20 OUTPUT 723 USING 10; "O00400T"

10 IMAGE #, A, 4D, A
20 OUTPUT 723 USING 10; "A",X,"T"
```

3. Data sent to the Multiprogrammer over the HP-IB must be in octal. See the attached routines for converting decimal to octal.

INPUT

A) 16-bit duplex I/O (98032A opt 040,340, or 440)

1. The following command may be used to send address words to the 6940B:

   WRITE IO 9,4; 12288

   The data, 12288, is the address word for slot3

2. The following command may be used to gate the input card and read back the data into variable A:

   A=READ BIN (9)

3. To read without gate, the following is used:

   READ IO 9, 4; A

4. For block transfer input this may be used:

   ENTER 9 type **NO FORMAT**; list

   type: **WHS**
   WINT
   WFHS count (count is the number of words to be read in)
   WDMA count

   list: the input variables, must be dimensioned as integers

B) HP-IB

1. The following is used to read data from the 6940B:

   ENTER 723, A

2. The following is used to read the 6940B status (to see if SRQ is set):

   STATUS 723; C

3. Data returned from the Multiprogrammer to the Desktop Computer over the HP-IB is in octal. See the attached routines for converting from octal to decimal.

Decimal To Octal Conversion

9845A/S

```
X= A+2*INT(A/8)+20*INT(A/64)+200*INT(A/512)+2000*INT(A/4096)+20000*INT(A/32768)+
(A<0)*200000
```

9835A/B, 9845B/T

```
X = OCTAL(A)
```

Octal To Decimal Conversion

9845A/S

```
X= A-2*INT(A/10)-16*INT(A/100)-128*INT(A/1000)-1024*INT(A/10000)-8192*INT(A/1000
00)-(A>77777)*65536
```

9835A/B, 9845B/T

```
X = DECIMAL(A)
```

DECIMAL CONTROL WORD CHART

Multiprogrammer Operating Mode		Control Word	
		OCTAL	DECIMAL
OUTPUT MODE	INPUT MODE		
ALL MODES OFF		170000	-4096
SYE		170040	-4064
DTE, SYE		170140	-4000
DTE, SYE, TME		170160	-3984
	ISL, SYE	170240	-3936
	ISL, SYE, TME	170260	-3920
	IEN, SYE, TME	170460	-3792

```
10  ! Talk Listen Test
20   OUTPUT 723
30   WAIT 200
40   ENTER 723
50   WAIT 200
60   GOTO 20
```

```
10   ! Data Output Test
20   IMAGE #,3A,4D,"X"
30   FOR N=0 TO 11
40   X=FNDto(2^N)
50   OUTPUT 723 USING 20;"OT@",X
60   WAIT 400
70   NEXT N
80   CALL Out("A")
90   CALL Out("B")
100  CALL Out("D")
110  CALL Out("H")
120  FOR N=0 TO 11
130  X=FNDto(4095-2^N)
140  OUTPUT 723 USING 20;"OTO",X
150  WAIT 400
160  NEXT N
170  CALL Out2("N7777")
180  CALL Out2("M7777")
190  CALL Out2("k7777")
200  CALL Out2("G7777")
210  GOTO 10
220  SUB Out(A$)
230  OUTPUT 723 USING "#,2A";A$,"X"
240  WAIT 400
250  SUBEND
260  DEF FNDto(A)
270      RETURN A+2*INT(A/8)+20*INT(A/64)+200*INT(A/512)+2000*INT(A/4096)+20000*
INT(A/32768)+(A(0)*200000
280  FNEND
290  SUB Out2(A$)
300  OUTPUT 723 USING "#,5A";A$,"X"
310  WAIT 400
320   SUBEND
```

```
10   ! Gate/Flag Test
20   OUTPUT 723 USING "#,6A";"O20T@T"
30   WAIT 400
40   OUTPUT 723 USING "#,A";"X"
50   WAIT 400
60   GOTO 20
```

```
10   ! Data Input Test
20   DISP "Put MP in REMOTE. Press CONTINUE."
30   PAUSE
40   OUTPUT 723 USING "#,4A";"OT@Z"
50   DISP "Throw MP from REMOTE to LOCAL"
60   WAIT 2000
70   ENTER 723;X
80   DISP X
90   GOTO 70
```

```
10   ! Serial Poll
20   OUTPUT 723 USING "#,4A";"O20T"
30   WAIT 500
40   STATUS 723;C
50   IF C=64 THEN 90
60   DISP "Status Error"
70   BEEP
80   PAUSE
90   WAIT 500
100  STATUS 723;C
110  IF C=0 THEN 30
120  DISP "SRQ Not Reset"
130  BEEP
140  PAUSE
150  GOTO 20
160 END
```

```
10   !       HP-IB EXAMPLE
20   !       DIGITAL OUTPUT
30   ! AUGUST 30, 1978              HEWLETT-PACKARD -- NJ DIV
40   ! TEST OF DIGITAL OUTPUT 6940B,9845 OR 9835
50   ! DIGITAL OUTPUT CARD IN SLOT 402 - B
60   ! SEE PAGE 5-12 IN THE 6940B -- 9825A USERS GUIDE FOR MORE INFO
70   !
80   INPUT "ENTER PIN NO.",A
90   IF A>12 THEN GOTO Error
100      A=FNDto(2^(A-1))
110  IMAGE #,6A,4D,A
120  OUTPUT 723 USING 110;"O140TA",A,"T"
130  DISP "THE BIT SHOULD BE SET, PRESS CONT TO CONTINUE"
140  PAUSE
150  IMAGE #,12A
160  OUTPUT 723 USING 150;"O160TBTO140T"
170  DISP "TEST COMPLETE"
180  STOP
190 Error:  DISP "PIN NO. MUST BE LESS THAN 13"
200  STOP
210  !
220  !
230  ! THIS IS THE DTO FUNCTION
240  !
250  DEF FNDto(A)
260      RETURN A+2*INT(A/8)+20*INT(A/64)+200*INT(A/512)+2000*INT(A/4096)+20000*
INT(A/32768)+(A<0)*200000
270  FNEND
280  END
```

```
10   !       HP-IB EXAMPLE
20   !       DIGITAL INPUT
30   ! AUGUST 30, 1978              HEWLETT-PACKARD -- NJ DIV
40   ! TEST OF DIGITAL INPUT 6940B,9845 OR 9835
50   ! DIGITAL INPUT CARD IN SLOT 403 - C
60   ! FOR MORE INFO SEE 6940B -- 9825A USERS GUIDE
70   !
80   IMAGE #,9A
90   OUTPUT 723 USING 80;"O260TCTCX"    ! Control Word and Address Word
100  ENTER 723;X
110  DISP X
120  STOP
130  END
```

```
10   ! Interface Card Address Test
20   IMAGE #,W
30   OUTPUT 9 WHS USING 20;-4096
40   WAIT 500
50   OUTPUT 9 WHS USING 20;4095
60   WAIT 500
70   GOTO 30


10   ! Output Data Test
20   FOR N=0 TO 14
30   WRITE IO 9,4;2^N
40   WAIT 500
50   NEXT N
60   WRITE IO 9,4;-32768
70   WAIT 500
80   FOR N=0 TO 14
90   WRITE IO 9,4;-1-2^N
100  WAIT 500
110  NEXT N
120  WRITE IO 9,4;32767
130  WAIT 500
140  GOTO 20


10   ! Gate/Flag Test
20   WRITE IO 9,5;32
30   WAIT 200
40   OUTPUT 9 WHS USING "#,W";-4080,0
50   WAIT 200
60   GOTO 20


10   ! Return Data Test
20   WRITE IO 9,5;0
30   DISP "Put MP in LOCAL. Press CONTINUE."
40   PAUSE
50   IF IOFLAG(9) THEN DISP "RETURN DATA is not pressed."
60   IF IOFLAG(9)=0 THEN DISP "RETURN DATA is pressed."
70   GOTO 50


10   ! Data Input Test
20   DISP "Put MP in REMOTE. Press CONTINUE."
30   PAUSE
40   WRITE IO 9,5;32
50   OUTPUT 9 WHS USING "#,W";-4096
60   DISP "Throw MP from REMOTE to LOCAL. Press CONTINUE. "
70   PAUSE
80   READ IO 9,4;A
90   IF A<0 THEN A=A+65536
100  A=A+2*INT(A/8)+20*INT(A/64)+200*INT(A/512)+2000*INT(A/4096)+20000*INT(A/32
768)
110  DISP A
120  GOTO 80
130  END
```

```
10 !            16-BIT DUPLEX EXAMPLE
20   !          DIGITAL OUTPUT
30   ! AUGUST 29, 1978          HEWLETT-PACKARD -- NJ DIV
40   ! TEST OF DIGITAL OUTPUT 6940B,9845 OR 9835
50   ! DIGITAL OUTPUT CARD IN SLOT 402
60   ! SEE PAGE 5-12 IN THE 6940B -- 9825A USERS GUIDE FOR MORE INFO
70   !
80   INPUT "ENTER PIN NO.",A
90   IF A>12 THEN GOTO Error
100      A=2^(A-1)
110      B=-4000  ! Control Word DTE, SYE
120      C=8192   ! Address Word slot 2
130  OUTPUT 9 WHS USING "#,W";B,C+A
140  DISP "THE BIT SHOULD BE SET, PRESS CONT TO CONTINUE"
150  PAUSE
160      D=-3984  ! Control Word DTE, SYE, TME
170  OUTPUT 9 WHS USING "#,W";D,C,B
180  DISP "TEST COMPLETE"
190  STOP
200 Error:  DISP "PIN NO. MUST BE LESS THAN 13"
210  STOP
220  END
```

```
10 !        16-BIT DUPLEX EXAMPLE
20   !      DIGITAL INPUT
30   ! AUGUST 29, 1978          HEWLETT-PACKARD -- NJ DIV
40   ! TEST OF DIGITAL INPUT 6940B,9845 OR 9835
50   ! DIGITAL INPUT CARD IN SLOT 403
60   ! SEE 6940B -- 9825A USERS GUIDE FOR MORE INFO
70   !
80      B=-3920   ! Control Word ISL, SYE, TME
90      C=12288   ! Address Word slot 3
100  OUTPUT 9 WHS USING "#,W";B
110  WRITE IO 9,4;C
120  A=READBIN(9)
130  PRINT "DATA = ";A+32768
140  STOP
150  END
```

```
10   !          16-BIT DUPLEX EXAMPLE
20   !          RELAY OUTPUT
30   ! MARCH 13, 1979                    HEWLETT-PACKARD -- NJD
40   !       69433A is assumed to be in Slot 2
50   !
60   Slot=8192    ! Slot 2
70 Mp_format:    IMAGE #,W
80          ! IMAGE specifies 16-BIT WORD and suppresses CR-LF
90          !
100 Again:  INPUT "Relay number? ( + to close, - to open)",Relay
110         ! Input the number of the relay to be changed
120         !
130  IF (ABS(Relay)<1) OR (ABS(Relay)>12) THEN GOTO Relay_error
140         ! Check for valid data
150         !
160  OUTPUT 9 WHS USING Mp_format;-3936 ! Control Word ISL, TME
170  WRITE IO 9,4;Slot       ! Address Word
180         !
190  READ IO 9,4;Relay_status
200         ! Read the 69433A relay status without sending a gate
210         !
220  IF Relay>0 THEN Data=BINIOR(Relay_status,ROTATE(1,1-Relay))
230         ! If 'Relay' is positive, close it without disturbing the others
240         !
250  IF Relay<0 THEN Data=BINAND(Relay_status,ROTATE(-2,1-ABS(Relay)))
260         ! If 'Relay' is negative, open it without disturbing the others
270         !
280  OUTPUT 9 WHS USING Mp_format;-4048,Slot+Data
290         ! Put the 6940B in the output mode (SYE,TME) and send the new data
300         !
310  GOTO Again
320         !
330 Relay_error:  DISP "Relay number must be between 1 and 12"
340         ! This is the error routine
350  BEEP
360  WAIT 2000
370  GOTO Again
380  END
```

```
10   !          16-BIT DUPLEX EXAMPLE
20   !          VOLTAGE OUTPUT
30   !   MARCH 13, 1979                    HEWLETT-PACKARD -- NJD
40   !   69321B is in Slot 2
50   !
60   Slot=8192  ! Slot 2
70 Mp_format:   IMAGE #,W
80         ! IMAGE specifies WORD and suppresses CR-LF
90         !
100 Again:  INPUT "Enter voltage? (-10.24 to +10.235V)",Voltage
110        ! Input the desired voltage
120        !
130  IF (Voltage<-10.24) OR (Voltage>10.235) THEN GOTO Voltage_error
140        ! Check for valid data
150        !
160  Data=Voltage/.005
170        ! Convert to the number of LSB's
180        !
190  IF Data<0 THEN Data=Data+4096
200        ! Check for negative value
210        !
220  OUTPUT 9 WHS USING Mp_format;-3984,Slot+Data
230        ! Put the 6940B in output mode (DTE,SYE,TME) and send the data
240        !
250  GOTO Again
260        !
270 Voltage_error:  DISP "Voltage is out of range"
280        ! This is the error routine
290        !
300  BEEP
310  WAIT 2000
320  GOTO Again
330  END
```

```
10    !      16-BIT DUPLEX EXAMPLE
20    !      VOLTAGE INPUT
30    ! MARCH 13, 1979                    HEWLETT-PACKARD -- NJD
40    !        69422A is assumed to be in Slot 2
50    !
60    Slot=8192      ! Slot 2
70 Mp_format:    IMAGE #,W
80           ! IMAGE specifies WORD and suppresses CR-LF
90           !
100   OUTPUT 9 WHS USING Mp_format;-3920  ! Control Word ISL, SYE, TME
110   WRITE IO 9,4;Slot  ! Address word without gate
120          !
130 Again:  Voltage_data=READBIN(9)
140          ! Read the 69422A sending a gate to trigger a new reading
150          !
160   IF Voltage_data<0 THEN Voltage_data=Voltage_data+32768
170          ! Check for IRQ and remove if present
180          !
190   IF Voltage_data>2047 THEN Voltage_data=Voltage_data-4096
200          ! Convert from 12-bit 2's complement
210          !
220   Voltage=Voltage_data*.005
230          ! Convert to volts
240          !
250   DISP "The measured voltage is";Voltage
260          ! Display the voltage
270          !
280   GOTO Again
290   END
```

IB321

D/A Voltage Converter Card, 69321B

(-10.24 to + 10.235 [illegible]

```
100  IMAGE #,K
110  INPUT "Desired output voltage?",Voltage
120  IF (Voltage<-10.24) OR (Voltage>10.235) THEN Error
130  V_code=OCTAL(Voltage/.005+(Voltage<0)*4096)
140  OUTPUT 723 USING 100;"O160TE",V_code,"T"
150  STOP
160 Error:  DISP "Voltage out of range"
170  END
```

IB330

```
100  IMAGE #,K
110  INPUT "Desired Relay output data",Relay
120  IF (Relay<0) OR (Relay>4095) THEN Error
130  R_code=OCTAL(Relay)
140  OUTPUT 723 USING 100;"O160TB",R_code,"T"
150  STOP
160 Error:  DISP "Relay output data out of range"
170  END
```

IB331

Digital Output Card, [illegible]

```
100  IMAGE #,K
110  INPUT "Desired Digital output data",Dig_out
120  IF (Dig_out<0) OR (Dig_out>4095) THEN Error
130  Do_code=OCTAL(Dig_out)
140  OUTPUT 723 USING 100;"O160TA",Do_code,"T"
150  STOP
160 Error:  DISP "Digital output data out of range"
170  END
```

IB332

```
100  IMAGE #,K
110  INPUT "Desired Open Collector output data?",Open_coll
120  IF (Open_coll<0) OR (Open_coll>4095) THEN Error
130  Oc_code=OCTAL(Open_coll)
140  OUTPUT 723 USING 100;"O40TB",Oc_code,"T"
150  STOP
160 Error:  DISP "Open Collector output data out of range"
170  END
```

IB335

```
100  IMAGE #,K
110  INPUT "Desired number of pulses (steps +=CW -=CCW)?",Pulses
120  IF (Pulses<-2047) OR (Pulses>2047) THEN Error
130  IF Pulses<0 THEN Pulses=2048+ABS(Pulses)
140  P_code=OCTAL(Pulses)
150  OUTPUT 723 USING 100;"O60TG",P_code,"T"
160  STOP
170 Error:  DISP "Number of pulses (steps) out of range"
180  END
```

IB370

```
100  IMAGE #,K
110  INPUT "Desired output current (in mA)?",Current
120  IF (Current<0) OR (Current>20.475) THEN Error
130  C_code=OCTAL(Current/.005)
140  OUTPUT 723 USING 100;"O160TE",C_code,"T"
150  STOP
160 Error:  DISP "Current out of range"
170  END
```

GP321

```
100  INPUT "Desired output voltage?",Voltage
110  IF (Voltage<-10.24) OR (Voltage>10.235) THEN Error
120  V_code=Voltage/.005+(Voltage<0)*4096
130  WRITE BIN 9;Cw+Dte+Sye+Tme,Slot5+V_code
140  STOP
150 Error:  DISP "Voltage out of range"
160  END
```

GP330

```
100  INPUT "Desired Relay data?",Relay
110  IF (Relay<0) OR (Relay>4095) THEN Error
120  WRITE BIN 9;Cw+Dte+Sye+Tme,Slot2+Relay
130  STOP
140 Error:  DISP "Relay data out of range"
150  END
```

GP331

```
100  INPUT "Desired Digital output data?",Dig_out
110  IF (Dig_out<0) OR (Dig_out>4095) THEN Error
120  WRITE BIN 9;Cw+Dte+Sye+Tme,Slot1+Dig_out
130  STOP
140 Error:  DISP "Digital output data out of range"
150  END
```

GP332

```
100  INPUT "Desired Open Collector output data?",Open_coll
110  IF (Open_coll<0) OR (Open_coll>4095) THEN Error
120  WRITE BIN 9;Cw+Sye,Slot2+Open_coll
130  STOP
140 Error:  DISP "Open Collector output data out of range"
150  END
```

GP335

```
100  INPUT "Desired number of pulses (steps +=CW -=CCW)?",Pulses
110  IF (Pulses<-2047) OR (Pulses>2047) THEN Error
120  IF Pulses<0 THEN Pulses=2048+ABS(Pulses)
130  WRITE BIN 9;Cw+Dte+Sye+Tme,Slot7+Pulses
140  STOP
150 Error:  DISP "Number of pulses out of range"
160  END
```

GP370

```
100  INPUT "Desired output current (in mA)?",Current
110  IF (Current<0) OR (Current>20.475) THEN Error
120  C_code=Current/.005
130  WRITE BIN 9;Cw+Dte+Sye+Tme,Slot5+C_code
140  STOP
150 Error:  DISP "Current out of range"
160  END
```

GP380

```
100  INPUT "Desired Breadboard output data?",Bread_out
110  IF (Bread_out<0) OR (Bread_out>4095) THEN Error
120  WRITE BIN 9;Cw+Sye,Slot2+Bread_out
130  STOP
140 Error:  DISP "Breadboard output data out of range"
150  END
```

GP422 A/D

```
100  WRITE BIN 9;Cw+Isl+Sye+Tme
110  WRITE IO 9,4;Slot6
120  V_code=READBIN(9)
130  IF V_code<0 THEN V_code=V_code+32768
140  Voltage=(V_code-(V_code>2047)*4096)*.005
150  DISP "Voltage =";Voltage
160  END
```

GP430

```
100  WRITE BIN 9;Cw+Isl+Sye
110  WRITE IO 9,4;Slot4
120  READ IO 9,4;Iso_in
130  DISP "Isolated Input data =";Iso_in
140  END
```

GP431

```
100  WRITE BIN 9;Cw+Isl+Sye+Tme
110  WRITE IO 9,4;Slot3
120  Dig_in=READBIN(9)
130  DISP "Digital Input data =";Dig_in
140  END
```

```
80   ! Example 1   Sending a Word to the Multiprogrammer
90   !
100  IMAGE #,K
110  OUTPUT 723 USING 100;Data_list
120  !
130  !
140  ! Example 2   Sending a Control Word
150  !
160  !
170  OUTPUT 723 USING 100;"0140T"
180  !
190  !
200  ! Example 3   Sending a Data Word
210  !
220  Dw=OCTAL(Data)
230  OUTPUT 723 USING 100;"0140TE",Dw,"T"
240  !
250  !
260  ! Example 4   Sending an Address Word with Gate
270  !
280  OUTPUT 723 USING 100;"0260TCT"
290  !
300  !
310  ! Example 5   Reading the Return Data Word
320  !
330  ENTER 723;Rdw
340  !
350  !
360  ! Example 6   Reading the Status Byte
370  !
380  STATUS 723;Srq
390  !
400  !
410  ! Example 7   Sending Constant and Variable Data Words
420  !
430  OUTPUT 723 USING 100;"A1425TE",Voltage,"T"
440  !
450  !
460  ! Example 8   SYE Parallel Initialization
470  !
480  OUTPUT 723 USING 100;"OTE",Init_volt,"TG100T040T"
490  !
500  !
510  ! Example 9   SYE System Shutdown
520  !
530 Shutdown:  OUTPUT 723 USING 100;"OT"
540  !
550  !
560  ! Example 10   DTE Parallel Output
570  !
580  OUTPUT 723 USING 100;"040TA",Data1,"TB",Data2,"TO140T"
590  !
600  !
610  ! Example 11   TME Serial Output
620  !
630  OUTPUT 723 USING 100;"0160TA",Data1,"TB",Data2,"TE",Data5,"T"
640  !
650  !
660  ! Example 12   TME Parallel Output
670  !
680  OUTPUT 723 USING 100;"0140TA",Data1,"TB",Data2,"TO160T"
690  !
700  !
710  ! Example 13   Waiting for Completion-Stacking Words
720  !
730  OUTPUT 723 USING 100;"0140TA",Data1,"TB",Data2,"TO160T040T"
```

```
740  !
750  !
760  ! Example 14   Waiting for Completion-Interrupt
770  !
780  STATUS 723;Srq
790  ON INT #7 GOSUB Isr_output
800  CONTROL MASK 7;128
810  OUTPUT 723 USING 100;"O140TE",Voltage,"TG",Position,"TO160T"
820  CARD ENABLE 7
830  !
840  !  User's Program Goes Here
850  !
860  STOP
870 Isr_output:  STATUS 723;Srq
880  IF Srq<>64 THEN Not_multi
890  PRINT "Multiprogrammer finished"
900  RETURN
910  !
920  !
930  !
940  ! Example 15   Testing for, and Removing IRQ
950  !
960  IF Rdw>7777 THEN Rdw=Rdw-10000
970  !
980  !
990  ! Example 16   Waiting for Completion, Input-Stacking Words
1000 !
1010 OUTPUT 723 USING 100;"O260TCTCX"
1020 ENTER 723;Rdw
1030 !
1040 !
1050 ! Example 17   Waiting for Completion, Input-Interrupt
1060 !
1070 ON INT #7 GOSUB Isr_input
1080 CONTROL MASK 7;128
1090 OUTPUT 723 USING 100;"O260T"
1100 STATUS 723;Srq
1110 OUTPUT 723 USING 100;"CT"
1120 CARD ENABLE 7
1130 !
1140 !  User's Program Goes Here
1150 !
1160 STOP
1170 Isr_input: STATUS 723;Srq
1180 IF Srq<>64 THEN Not_multi
1190 ENTER 723;Rdw
1200 PRINT "Data=";Rdw
1210 RETURN
1220 !
1230 !
1240 !
1250 ! Example 18   Address and Read without Gate
1260 OUTPUT 723 USING 100;"O240THX"
1270 ENTER 723;Rdw
1280 !
1290 !
1300 PAUSE
1310 ! Example 19   Parallel Input IEN Mode
1320 !
1330 ON INT #7 GOSUB Isr_ien
1340 CONTROL MASK 7;128
1350 STATUS 723;Srq
1360 OUTPUT 723 USING 100;"O240TCTDT0460T"
1370 CARD ENABLE 7
1380 !
1390 !  User's Program Goes Here
```

```
1400 !
1410 STOP
1420 Isr_ien: STATUS 723;Srq
1430 IF Srq<>64 THEN Not_multi
1440 OUTPUT 723 USING 100;"0240TCX"
1450 ENTER 723;Rdw
1460 IF Rdw<10000 THEN Next_card
1470 Data3=Rdw-10000
1480 PRINT "Slot # generated interrupt, data=";Data3
1490 OUTPUT 723 USING 100;"CT"
1500 Next_card: OUTPUT 723 USING 100;"DX"
1510 ENTER 723;Rdw
1520 IF Rdw<10000 THEN Done
1530 Data4=Rdw-10000
1540 PRINT "Slot 4 generated interrupt, data=";Data4
1550 OUTPUT 723 USING 100;"DT"
1560 Done: OUTPUT 723 USING 100;"0460T"
1570 CARD ENABLE 7
1580 RETURN
1590 !
1600 !
1610 !
1620 ! Example 20   Leaving IEN Mode
1630 !
1640 Leave_ien: OFF INT #7
1650 OUTPUT 723 USING 100;"X040T"
1660 !
1670 !  User's Program Goes Here
1680 !
1690 ON INT #7 GOSUB Isr_ien
1700 STATUS 723;Srq
1710 OUTPUT 723 USING 100;"0460T"
1720 CARD ENABLE 7
1730 RETURN
1740 !
1750 !
```

```
1820 Data3=Rdw+32768
1830 PRINT "SLOT3 GENERATED INTERRUPT, DATA=";Data3
1840 WRITE BIN 9;Slot3
1850 Next_card: WRITE IO 9,4;Slot4
1860 READ IO 9,4;Rdw
1870 IF Rdw>=0 THEN Done
1880 Data4=Rdw+32768
1890 PRINT "SLOT4 GENERATED INTERRUPT, DATA=";Data4
1900 WRITE BIN 9;Cw+Sye,Slot4
1910 Done: WRITE BIN 9;Cw+Ien+Sye+Tme
1920 CARD ENABLE 9
1930 RETURN
1940 !
1950 !
1960 !
1970 ! Example 20   Leaving IEN Mode
1980 !
1990 Leave_ien: OFF INT #9
2000 RESET 9
2010 WRITE BIN 9;Cw+Sye
2020 !
2030 !  User's Program Goes Here
2040 !
2050 ON INT #9 GOSUB Isr_ien_mode
2060 WRITE BIN 9;Cw+Ien+Sye+Tme
2070 CARD ENABLE 9
2080 RETURN
```

•IB321

```
100  IMAGE #,K
110  INPUT "Desired output voltage?",Voltage
120  IF (Voltage<-10.24) OR (Voltage>10.235) THEN Error
130  V_code=OCTAL(Voltage/.005+(Voltage<0)*4096)
140  OUTPUT 723 USING 100;"0160TE",V_code,"T"
150  STOP
160 Error:  DISP "Voltage out of range"
170  END
```

IB330

```
100  IMAGE #,K
110  INPUT "Desired Relay output data",Relay
120  IF (Relay<0) OR (Relay>4095) THEN Error
130  R_code=OCTAL(Relay)
140  OUTPUT 723 USING 100;"0160TB",R_code,"T"
150  STOP
160 Error:  DISP "Relay output data out of range"
170  END
```

IB331

```
100  IMAGE #,K
110  INPUT "Desired Digital output data",Dig_out
120  IF (Dig_out<0) OR (Dig_out>4095) THEN Error
130  Do_code=OCTAL(Dig_out)
140  OUTPUT 723 USING 100;"0160TA",Do_code,"T"
150  STOP
160 Error:  DISP "Digital output data out of range"
170  END
```

IB332

```
100  IMAGE #,K
110  INPUT "Desired Open Collector output data?",Open_coll
120  IF (Open_coll<0) OR (Open_coll>4095) THEN Error
130  Oc_code=OCTAL(Open_coll)
140  OUTPUT 723 USING 100;"040TB",Oc_code,"T"
150  STOP
160 Error:  DISP "Open Collector output data out of range"
170  END
```

IB335

```
100  IMAGE #,K
110  INPUT "Desired number of pulses (steps +=CW -=CCW)?",Pulses
120  IF (Pulses<-2047) OR (Pulses>2047) THEN Error
130  IF Pulses<0 THEN Pulses=2048+ABS(Pulses)
140  P_code=OCTAL(Pulses)
150  OUTPUT 723 USING 100;"060TG",P_code,"T"
160  STOP
170 Error:  DISP "Number of pulses (steps) out of range"
180  END
```

IB370

```
100  IMAGE #,K
110  INPUT "Desired output current (in mA)?",Current
120  IF (Current<0) OR (Current>20.475) THEN Error
130  C_code=OCTAL(Current/.005)
140  OUTPUT 723 USING 100;"0160TE",C_code,"T"
150  STOP
160 Error:  DISP "Current out of range"
170  END
```

INTRODUCTION

1-1 SCOPE

1-2 This guide provides programming, installation, and verification procedures for the 6940B Multiprogrammer system as controlled by an HP 9835A or 9845B Desktop Computer using either the HP 98032A 16-bit Duplex Interface, Option 340 (for the 9835A) or Option 440 (for the 9845B), or the HP 98034A HP-IB Interface. The 16-bit Duplex Interface provides a 16-bit word directly to the 6940B while the HP-IB Interface provides an 8-bit ASCII coded output which must be converted to the 16-bit format required by the 6940B. Thus, when the 98034A HP-IB interface is used, a 59500A HP-IB to Multiprogrammer Interface unit must be employed to convert the ASCII code to the 6940B format. This guide includes condensed circuit desciptions of the 6940B/6941B Multiprogrammer (Chapter VII) and the 59500A Multiprogrammer Interface (Chapter VIII). Although these descriptions do not provide a complete hardware analysis showing every signal and circuit, they are quite comprehensive and provide the user with a good understanding of the Multiprogrammer system capabilities. The intent of this guide, therefore, is to provide all of the information to program and use the 6940B Multiprogrammer with the 9835A or 9845B Desktop Computer and minimize the need for constant referrals to other documents (manuals, data sheets, etc.).

1-3 The following is an abbreviated listing of all of the major information contained in this guide, together with a brief description concerning the scope of the material.

(1) Installation and Checkout. Chapter II contains all of the information necessary for interconnecting the units, setting the address switches, and turning on the equipment in the proper sequence. Checkout procedures are included to verify that the equipment is connected correctly and that the major functions of the Multiprogrammer system can be programmed by a 9835A or 9845B Desktop Computer. Verification program listings are provided for both the 16-bit Duplex interface and the HP-IB interface. The programs verify operation of the 6940B mainframe (and the 59500A unit when using the HP-IB intrface) excluding plug-in cards and 6941B extender units. The verification programs are essentiolly "GO/NO-GO" checks, for more complete hardware diagnosis the @ @ @
sevice kit info goes here @

(2) Programming Fundamentals. Chapter III outlines the fundamentals of programming a 6940B/6941B Multiprogrammer system with a 9835A or 9845B Desktop Computer using the 98032A 16-bit Duplex Interface, while Chapter IV outlines the fundamentals of programming a 59500A/6940B/6941B Multiprogrammer system with a 9835A or 9845B Desktop Computer using the 98034A HP-IB Interface. In addition, Chapter VII Paragraphs 7-xx through 7-xx, include a theoretical explanation of the 6940B programming sequences, complete with flow diagrams. Note that the information in this guide relates only to programming requirements that are unique to the Multiprogrammer system. It is assumed that the user is familior with the fundamentals of programming a 9835A or 9845B Desktop Computer.

(3) Plug-In Card Descriptions and Programs. Chapter V

contains a brief functional description for each type of Multiprogrammer plug-in card. In addition, both 16-bit Duplex and HP-IB interface programming examples are provided for each card type.

(4) Other Programming Techniques. Chapter VI contains programming hints to aid the user in making the programs self-documenting and easier to modify. Also, some advanced I/O techniques that can be used with the 6940B Multiprogammer system are discussed.

(5) 6940B and 59500A Circuit Descriptions. As mentioned previously, simplified circuit descriptions of these two units are given in Chapters VII and VIII. Users who are not familiar with the Multiprogrammer or who have complex programmming requirements are strongly urged to read these two chapters. The 59500A description (Chapter VIII) is pertinent only to operating the Multiprogrammer on the HP-IB.

(6) Number System Theory and Utility Subroutines. Appendix A contains descriptions of the number systems applicable to the Multiprogrammer system. Included are explantations of the decimal, octal, binary, and BCD systems and how to convert from one system to another. Appendix A also contains utility subroutines which are addtionsl programming aids for the user. The subroutines can be called from the user's main program to perform required operations.

1-4 RELATED PUBLICATIONS

1-5 Other Multiprogrammer literature that will be helpful to the user, includes the Operating and Service Manuals for the Multiprogrammer mainframes (6940B and 6941B), the HP-IB to Multiprogrammer Interface unit (59500A), and for each type of plug-in card. Each manual contains troubleshooting instructions, circuit theory, circuty diagrams and a replaceable parts list. Information on a more general level, such as card descriptions and specifications, is contained int the 6940B Multiprogrammer bochure (5952-xxxx).

1-6 Publicattions for the 9835A and 9845B that the user may find useful include the following:

computer manual numbers and
descriptions go here

CHAPTER II

INSTALLATION AND CHECKOUT

2-1 This chapter provides procedures for interconnecting the 6940B/6941B Multiprogrammer system with a 9835A or 9845B Desktop Computer using either the 98032A 16-bit Duplex Interface or the 98034A HP-IB Interface. The installation procedures include gathering the proper equipment, installing the applicable ROM's, setting address switches, and connecting the cables. Checkout procedures are included to verify that the equipment is installed correctly and that the Multiprogrammer system can be programmed properly by the 9835A or 9845B Desktop Computer. Paragraphs 2-2 through 2-18 provide installation and checkout procedures for the 16-bit Duplex interface, while paragraphs 2-19 through 2-40 provide similar coverage for the HP-IB interface.

2-2 16-BIT DUPLEX INTERFACE

2-3 Equipment Required

2-4 The following equipment is required to assemble the system:

1. HP 6940B Multiprogrammer (plus 6941B Multiprogrammer Extender units if required).
2. Input/Output plug-in cards for the 6940B/6941B Multiprogrammer system.
3. HP 9835A or 9845B Desktop Computer.
4. HP 98032A 16-bit Duplex Interface, Option 340 (for the 9835A) or Option 440 (for the 9845B).
5. HP 98332A I/O ROM (for the 9835A) or HP 98432A I/O ROM (for the 9845B).
6. HP 14541A Chaining Cable, only if 6941B extender units are used (must be ordered separately). Standard length is 0.46 meters (18 inches), lengths up to 30 meters (100 feet) are available on special order.

2-5 9835A I/O ROM and 98032A-Option 340 Installation

2-6 The 9835A has four ROM drawers located in the bottom front of the computer. Each ROM drawer has four slots into which the ROM's can be inserted. The 98332A I/O ROM consists of 4 parts which can be plugged into any of the available slots. The power must be turned off before attempting to install the ROM's or damage may result. For more detailed information, consult the 9835A I/O ROM manual (09835-90060).

2-7 The 98032A-Option 340 16-bit Duplex Interface is configured to operate with the 6940B Multiprogrammer and can be installed in any of the three I/O slots in the rear of the 9835A Desktop Computer. Again, the power must be off before installing the 98032A or damage may result. The select code for the 98032A-Option 340 is preset to "9" which is the code assigned to the Multiprogrammer system. Throughout this guide, all examples and sample programs for the 16-bit Duplex Interface use a select code of 9. To change the select code, the small rotary switch at the cable end of the 98032A is adjusted to the desired select code. For more detailed information, consult the 98032A Installation and Programming Manual (98032-90000).

2-8 9845B I/O ROM and 98032A-Option 440 Installation

2-9 The 9845B has two ROM drawers located on the sides of the computer. Each ROM drawer has a number of slots labeled for option ROM's. The 98432A I/O ROM consists of 2 parts, one part for each drawer, which must be installed in the correct drawer as determined by color and guide pins. The power must be turned off before attempting to install the ROM's, or damage may result. For more detailed information, consult the 9845B I/O ROM Manual (09845-91060).

2-10 The 98032A-Option 440 16-bit Duplex Interface is configured to operate with the 6940B Multiprogrammer and can be installed in any of the four I/O slots in the rear of the 9845B Desktop Computer. Again, the power must be off before installing the 98032A or damage may result. The select code for the 98032A-Option 440 is preset to "9", which is the code assigned to the Multiprogrammer system. Throughout this guide all examples and sample programs for the 16-bit Duplex Interface use a select code of 9. To change the select code, the small rotary-switch at the cable end of the 98032A is adjusted to the desired select code. For more detailed information, consult the 98032A Installation and Programming Manual (98032-90000).

2-11 Connecting Cables

2-12 Connect cables as shown in Figure 2-1. The 98032A Option 340/440 interface cable is equipped with the proper connector to mate with the J1 DATA INPUT connector on the rear of the 6940B Multiprogrammer. One 14541A Chaining Cable is required for each 6941B Multiprogrammer Extender unit used in the system. The standard length of the 14541A cable is 0.46 meters (18 inches) with cables up to 30 meters (100 feet) available on special order.

2-13 Pre-Operational Checklist

2-14 Run through the following checklist before programming the system.

1. Correct AC line voltage is selected on the rear of the 6940B and 6941B (if used). See Chapter 2 in 06940-90005 and 06941-90002 Operating and Service manuals.
2. Correct AC line voltage is selected on the 9835A or 9845B as described in the Owner's Manuals, HP Nos. 09835-90005 or 09845-91005.
3. 98032A Interface is installed with select code "9" selected.
4. I/O ROM's are properly installed.
5. If desired, verify that the 6940B (excluding all plug-in cards and 6941B Extenders) is operating properly by performing the checkout procedures given in Paragraphs 2-15 through 2-18.
6. Required input/output plug-in cards are installed in the 6940B and 6941B (if used) mainframe(s). Make a list of all the plug-in cards with their mainframe number and slot address. This list will be useful when writing and debugging programs.

----- CAUTION -----

Ensure that power is turned off at the 6940B and 6941B (if used) when inserting or removing cards. Failure to do so may result in damage to the card and/or Multiprogrammer mainframe.

7. Cables are properly connected (see Figure 2-1).
8. Power is applied to the 9835A or 9845B, 6940B, and 6941B (if used). Ensure that all 6941B's connected in the system are turned on.

2-15 Checkout Procedures

2-16 The following procedures verify that the major functions of the 6940B Multiprogrammer can be programmed from a 9835A or 9845B Desktop Computer using the 16-bit Duplex Interface. The programs are simple "GO/NO-GO" checks to establish that the system is operational. Complete hardware diagnosis of inoperable 6940B functions is provided in Operating and Service Manual HP Part No. 06940-90005; and

service kit goes here

The verification programs are designed to exercise the 6940B only; do not install any plug-in cards in the 6940B mainframe. Also, do not connect a 6941B Extender to the 6940B when performing the checkout procedures.

2-17 Use the following test setup for the 16-bit Duplex verification tests.

1. Open the 6940B front door and remove all input/output plug-in cards from mainframe slots 400 through 414.

----- CAUTION -----

Ensure that power is turned off at the 6940B when removing plug-in cards. Failure to do so may result in damage to the plug-in cards and/or the mainframe.

2. Ensure that the A1 Standard Input, A2 Remote/Local, A3 Logic and Timing, and A5 Unit Select cards are installed in mainframe slots 100, 200, 300, and 500 respectively. These cards are clearly marked and must be installed to check the operation of the 6940B.
3. Connect cables as shown in Figure 2-1, except do not connect a 6941B to the 6940B using the 14541A cable.

2-18 Verification Tests. It is recommended that the verification tests be executed in the order presented the first time the 6940B Multiprogrammer system is placed in operation. Later, after the system has been operational, the user can select any of the tests to verify functions that are suspected of being faulty. Each test contains operating instructions, a listing of the test program, and the test results (indications on the 6940B front panel and/or computer display). If the test results are not as specified, the user should clear up the problem and rerun the test before continuing with subsequent verification tests. While the failure of a test may indicate that a hardware problem

exists, certain operational or equipment configuration mistakes can be made that will result in erroneous indications. In the case of an incorrect indication, ensure that the test setup is correct and that the verification program was keyed in correctly.

2-19 HP-IB INTERFACE

2-20 Equipment Required

2-21 The following equipment is required to assemble the system:

1. HP 6940B Multiprogrammer (plus 6941B Multiprogrammer Extender units if required).
2. HP 59500A Multiprogrammer Interface.
3. Input/Output plug-in cards for the 6940B/6941B Multiprogrammer system.
4. HP 9835A or 9845B Desktop Computer.
5. HP 98034A HP-IB Interface.
6. HP 98332A I/O ROM (for the 9835A) or HP 98432A I/O ROM (for the 9845B).
7. HP 14541A Chaining Cable, only if 6941B extender units are used (must be ordered separately). Standard length is 0.46 meters (18 inches), lengths up to 30 meters (100 feet) are available on special order.

2-22 9835A I/O ROM and 98034A Installation

2-23 The 9835A has four ROM drawers located in the bottom front of the computer. Each ROM drawer has four slots into which the ROM's can be inserted. The 98332A I/O ROM consists of 4 parts which can be plugged into any of the available slots. The power must be turned off before attempting to install the ROM's or damage may result. For more detailed information, consult the 9835A I/O ROM manual (09835-90060).

2-24 The 98034A HP-IB Interface provides HP-IB control capability for the 9835A Desktop Computer and can be installed in any of the three I/O slots in the rear of the 9835A. Again, power must be off before installing the 98034A or damage may result. The select code for the 98034A is preset to "7". Throughout this guide, all examples and sample programs for the HP-IB Interface use a select code of 7. To change the select code, the small rotary switch at the cable end of the 98034A is adjusted to the desired select code. For more detailed information, consult the 98034A Installation and Programming Manual (98034-90000).

2-25 9845B I/O ROM and 98034A Installation

2-26 The 9845B has two ROM drawers located on the sides of the computer. Each ROM drawer has a number of slots labeled for option ROM's. The 98432A I/O ROM consists of 2 parts, one part for each drawer, which must be installed in the correct drawer as determined by color and guide pins. The power must be turned off before attempting to install the ROM's, or damage may result. For more detailed information, consult the 9845B I/O ROM Manual (09845-90000).

2-27 The 98034A HP-IB Interface provides HP-IB control capability for the 9845B Desktop Computer and can be installed

any of the four I/O slots in the rear of the 9845B. Again, the power must be off before installing the 98034A or damage may result. The select code for the 98034A is preset to "7". Throughout this guide all examples and sample programs for the HP-IB Interface use a select code of 7. To change the select code, the small rotary switch at the cable end of the 98034A is adjusted to the desired select code. For more detailed information, consult the 98034A Installation and Programming Manual (98034-90000).

2-28 Setting Multiprogrammer System Addresses

2-29 The talk and listen addresses for the Multiprogrammer system are selected by address switches on the rear of the 59500A Multiprogrammer Interface unit. The switches are set to the suggested talk address of "W" and listen address of "7" as the unit is shipped from the factory. The "W" and "7" correspond to an address of "23" as defined in the I/O ROM Manuals for the 9835A and 9845B Desktop Computers. Throughout this guide, all examples and sample programs for the HP-IB Interface use a Multiprogrammer address of "23".

2-30 As shown in Figure 2-3, there are seven address switches on the rear of the 59500A. The last two switches, 6 and 7 are ignored. Check that switches 1 through 3 and 5 are set to "1" and switch 4 is set to "0" to select a talk address of"W" and a listen address of "7".

2-31 Connecting Cables

2-32 Connect cables as shown in Figure 2-4. The 98034A HP-IB Interface cable is equipped with the proper connector to mate with the J1 HP-IB connector on the rear of the 59500A Multiprogrammer Interface. The 14541A Chaining Cable connecting the 59500A to the 6940B is supplied with the 59500A. One 14541A Chaining Cable is required for each 6941B Multiprogrammer Extender unit used in the system. The standard length of the 14541A cable is 0.46 meters (18 inches) with cables up to 30 meters (100 feet) available on special order.

2-33 Pre-Operational Checklist

2-34 Run through the following checklist before programming the system.

1. Correct AC line voltage is selected on the rear of the 59500A, 6940B and 6941B (if used). See Chapter 2 in 59500-90001, 06940-90005, and 06941-5000: Operating and Service Manuals.
2. Correct AC line voltage is selected on the 9835A or 9845B as described in the Owner's Manual, HP Nos. 09835-90005 or 09845-91005.
3. 98034A Interface is installed with select code "7" selected.
4. Address switches on the rear of the 59500A are set for a Multiprogrammer talk address of "W" and a listen address of "7" (see Paragraph 2-30).
5. I/O ROM's are properly installed.
6. If desired, verify that the 59500A/6940B (excluding all plug-in cards and 6941B Extenders) is operating properl

by performing the checkout procedures given in Paragraphs 2-37 through 2-40.

7. Required input/output plug-in cards are installed in the 6940B and 6941B (if used) mainframe(s). Make a list of all the plug-in cards with their mainframe number and slot address. This list will be useful when writing and debugging programs.

----- CAUTION -----

Ensure that power is turned off at the 6940B and 6941B (if used) when inserting or removing cards. Failure to do so may result in damage to the card and/or Multiprogrammer mainframe.

8. Cables are properly connected (see Figure 2-4).

2-35 System Turn-On Sequence

2-36 To prevent the random setting of the 59500A's service request (SRQ) and the Multiprogrammer's system enable signal (SYE) at turn-on, the instruments must be turned on in the following sequence:

1. Turn on the computer (and peripherals).
2. Turn on the 6940B and set it to LOCAL.
3. Turn on the 59500A and 6941B's (if used). Ensure that all 6941B's connected in the system are turned on.
4. Now switch the 6940B to REMOTE.

2-37 Checkout Procedures

2-38 The following procedures verify that the major functions of the 59500A/6940B Multiprogrammer system can be programmed from a 9835A or 9845B Desktop Computer using the HP-IB Interface. The programs are simple "GO/NO-GO" checks to establish that the system is operational. Complete hardware diagnosis of inoperable 6940B functions is provided in Operating and Service Manuals for the 59500A and 6940B.

service kit goes here

The verification programs are designed to exercise the 59500A/6940B only; do not install any plug-in cards in the 6940B mainframe. Also, do not connect a 6941B Extender to the 6940B when performing the checkout procedures.

2-39 Use the following test setup for the HP-IB verification tests.

1. Open the 6940B front door and remove all input/output plug-in cards from mainframe slots 400 through 414.

----- CAUTION -----

Ensure that power is turned off at the 6940B when removing plug-in cards. Failure to do so may result in damage to the plug-in cards and/or the mainframe.

2. Ensure that the A1 Standard Input, A2 Remote/Local, A3 Logic and Timing, and A5 Unit Select cards are installed in mainframe slots 100, 200, 300, and 500 respectively. These cards are clearly marked and must be installed to check the operation of the 6940B.
3. Connect cables as shown in Figure 2-4, except do not connect a 6941B to the 6940B using the 14541A cable.

2-40 Verification Tests. It is recommended that the verification tests be executed in the order presented the first time the 59500A/6940B Multiprogrammer system is placed in operation. Later, after the system has been operational, the user can select any of the tests to verify functions that are suspected of being faulty. Each test contains operating instructions, a listing of the test program, and the test results (indications on the 59500A or 6940B front panel and/or computer display). If the test results are not as specified, the user should clear up the problem and rerun the test before continuing with subsequent verification tests. While the failure of a test may indicate that a hardware problem exists, certain operational or equipment configuration mistakes can be made and result in erroneous indications. In the case of an incorrect indication, ensure that the test setup is correct and that the verification program was keyed in correctly.

CHAPTER III

16-BIT DUPLEX PROGRAMMING FUNDAMENTALS

3-1 This chapter outlines the fundamentals of programming the Multiprogrammer system with the 9835A or 9845B Desktop Computer using the 98032A 16-bit Duplex Interface (Option 340 for 9835A or Option 440 for 9845B). Included in this chapter are explanations and examples showing how to program the basic word formats, as well as general programming techniques for Multiprogrammer input and output cards. All examples assume the 16-bit Duplex interface is assigned a select code of 9.

NOTE

3-2 MULTIPROGRAMMER WORD FORMATS

3-3 The next portion of this chapter is devoted to the fundamentals of communicating between the Desktop Computer and the Multiprogrammer system. For computer to Multiprogrammer exchanges, three different types of words can be sent, and for Multiprogrammer to computer exchanges one type of word is received. The following paragraphs describe the basic construction of each word type, showing how to send the output words or read in the input word. Each word type is treated separately without too much regard for the programming sequences which can be used to accomplish a specific task. Programming examples, using these basic word formats, are included later in this chapter, in the General Programming Techniques sections.

3-4 Sending Control, Data, and Address Words

3-5 The BASIC language instruction used to send a word to the Multiprogrammer is illustrated in Example 1. The command "WRITE BIN" writes in a binary form, to the interface set to select code "9", the 16-bit Duplex interface, the value of variable "Cw". Note that a constant may be used instead of a variable.

3-6 Mnemonic Word Constuction. A mnemonic technique will be used throughout this guide to construct the word formats that are required for the computer to Multiprogrammer exchanges. A set of variables, that can be added together to yield the correct values for the word type being sent, as well as desribe the word type, are defined in the beginning of the program. The variable- that will be used are defined in Tables 3-1 and 3-2. Only those variables that are required in the user's system need be entered, for example, if there are no 6941B Extenders in the user's system, the variables defining the Unit addresses need not be entered. A complete desription of how to use Tables 3-1 and 3-2 is contained in the following paragraphs.

3-7 Control Word. The first word sent to a Multiprogrammer system is a contol word. It is used to specify a unit address and the operating mode of the Multiprogrammer system. The unit address portion selects which mainframe in the chain (of one

...)

6940B Multiprogrammer and up to fifteen 6941B Extenders) will respond to subsequent data or address words from the computer. The mode portion of the word selects one of several output or input modes of Multiprogrammer operation (System Enable, Timing Mode Enable, Input Select, etc.). The mode selected affects the entire Multiprogrammer system, including all 6941B Extender units. How the Multiprogrammer system responds to these various control word modes is described in Chapter VII of this guide, as well as in the 6940B Operating and Service Manual. A thorough understanding of these modes is essential for effective programming involving specific applications.

3-8 Figure 3-1 illustrates the structure of a control word as transmitted by the 16-bit Duplex interface. The four most significant bits are set to ones indicating to the 6940B that the remaining twelve bits are to be interperted as a control word. The mnemonic that is used to specify a control word is "Cw", this will cause the four most significant bits of the 16 bits sent to the Multiprogrammer to be set to ones.

3-9 The data field, consisting of the lower 12 bits, contains two pieces of information. The operating mode of the Multiprogrammer system is contained in bits 4 through 8. The various operating mode mnemonics (Ien, Isl, Ste, Sye, and Tme) are added to the "Cw" control word identifier to set the appropriate bits. The unit address of the mainframe that subsequent transfers will be made with is contained in bits 0 through 3. The unit address is determined by the physical position of the unit in the chain. The first unit in the chain, the 6940B, has a unit address of 0, the second unit, a 6941B, has an address of 1, and so forth, up to a unit address of 15. The unit address specifier, "UnitX", where "X" is the unit number, is simply added to the other control word mnemonics to complete the control word. Note that Unit0 need not be included except for documentation purposes. Bits 9 through 11 are always zero in a control word.

3-10 The construction of a typical control word with DTE and SYE on, addressing unit 3 is shown in Example 2 along with the BASIC instruction, WRITE BIN, that will send it to the Multiprogrammer. The WRITE BIN will send the control word with a gate, this is required to store the control word in the Multiprogrammer

3-11 Data Word. Data words select and program output cards and some functions of a few input cards. The data word selects an individual card, within the mainframe previously specified by a control word, and sends a 12-bit value to it.

3-12 Figure 3-2 shows the construction of the data word as sent to the Multiprogrammer by the 16-bit Duplex interface. The four most significant bits contain the address of the card that the remaining 12 bits of data will be transferred to. The card address portion has a range of 0 to 14, while the data portion has a range of 0 to 4095. (The slots for the I/O cards, 400 through 414, will be referred to as slots 0 through 14 in this guide.) To construct a data word, the mnemonic corresponding to the card slot is selected from Table 3-2, and the data that is to be sent to the card is added to it. The data portion must be in the correct range (0 to 4095) and format for the card being programmed as described in Chapter V. If the data is outside of

this range, it will overflow into the address portion of the data word causing the wrong card to be programmed.

3-13 Example 3 illustrates how to send a data word to a card in slot 5, unit 3. The control word in the previous example is also sent to select unit 3 and to program the output mode of operation for the Multiprogrammer system. The variable "Data" contains the properly formatted data for the card being programmed. Note that a constant can be used if the data will never change. The WRITE BIN statement will send the data word with a gate, this will cause the card addressed to store the data and initiate its action.

3-14 Address Word. An address word selects the input card that is to send data (a return data word) back to the computer. Before an address word can be sent, a control word must have first selected an input operating mode (with ISL on) and the unit address of the mainframe that contains the card.

3-15 Figure 3-3 shows the construction of an address word as sent to the Multiprogrammer by the 16-bit Duplex interface. The four most significant bits select the card that is to send data back to the computer. The remaining 12 bits are all set to zero. To send an address word, only the mnemonic corresponding to the desired card address, as selected from Table 3-2, need be sent.

3-16 Example 4 shows one of the techniques that can be used to send an address word, for slot 6, to the Multiprogrammer. This example, using the WRITE BIN instruction, will send the address word with a gate. Notice that a control word with ISL mode selected is sent prior to the address word.

3-17 Example 5 shows the the same address word being sent without a gate using the WRITE IO command. The "4" in the WRITE IO command selects the primary data register of the interface. For a more complete discussion of the interface registers than will be presented in this guide, the reader is referred to "BASIC Language Interfacing Concepts" (09835-90600). Notice that a control word with ISL on is also sent in this example using the WRITE BIN instruction. A discussion of when to send an address word with or without a gate is presented later.

3-18 Decimal Representations. In the preceding discussion a mnemonic technique was presented for the construction of Control, Data, and Address words; the method used to obtain the values used will be shown in this section.

3-19 All of the transfers made between the computer and the Multiprogrammer are 16-bit binary transfers. The computer deals with the numbers in these transfers as 16-bit two's complement values (see Appendix A for a discussion of two's complement) with a decimal range of -32768 to 32767, therefore all the values sent to the Multiprogrammer must be in this range. The decimal values that are required to program a control word can be determined by converting the corresponding 16-bit word (as shown in Figure 3-1) into its equivalent decimal value. Common control word values are given in Table 3-3, others can be calculated by adding the required modes from Table 3-1.

3-20 The decimal address portion of a data word and an address

word can be calculated by converting the 16-bit value as determined by the construction of the data word illustrated in Figure 3-2, or by the following equation:

Decimal address=Slot No.*4096-(Slot No.)7)*65536

This equation yields the correct value for the address portion of a data word. The data for the addressed card is then simply added to the decimal address.

3-21 Reading the Return Data Word

3-22 Addressing a card, while in the ISL mode, will allow the card to send its data back to the 16-bit Duplex interface. In general, the method used to read the data into the computer depends on the method used to address the card. If the card was addressed with a gate, as shown in Example 4, the computer's READ IO statement is used to read the data. The READBIN statement is used to read the data from the card if it was addressed without a gate, as in Example 5. Example 6a shows how to read the data if the card was addressed with a gate, Example 6b shows how to read the data if the card was addressed without a gate. In both cases the data is stored in the variable "Rdw". Exceptions to this rule will be presented with the applicable cards in Chapter V.

3-23 Words sent from the Multiprogrammer to the computer are known as return data words. The return data word contains information from an input card previously selected by an address word. Each word contains two pieces of information (see Figure 3-5); an input request (IRQ) bit and 12 bits of data.

3-24 The IRQ bit can be either a "1" or a "0", this translates to a decimal value of -32768 or 0. Thus, if the data stored in the computer is negative, IRQ was present. IRQ was not returned if the data is postitve. Not all of the input/output cards can return IRQ, refer to Table 3-4 for a list of the cards that have IRQ capability.

3-25 The main use of IRQ is in the IEN mode of operation when a "1" (negative data) indicates that the card has generated an interrupt and a "0" (positve data) indicates the card did not generate an interrupt. However, a "1" also indicates the input card has returned a flag and the data is valid, while a "0" may indicate that a) the card has not returned a flag (it is still busy), or b) the card does not have IRQ capability.

3-26 The lower 12 bits contain the data that the input card is sending to the computer. The range of the data will be between 0 and 4095. Note that the IRQ bit is not part of the data and if it is present it must be removed from the data prior to calculations. Testing for and removing IRQ will be described later in the general programming techniques section.

3-27 GENERAL PROGRAMMING INFORMATION

3-28 The next portion of this chapter describes the general programming sequences associated with the various operating modes and plug-in cards of the Multiprogrammer system. If a detailed description of Multiprogrammer operation is required, see Chapter

VII. For specific examples relating to a plug-in card, see Chapter V.

3-29 Number System Theory. Appendix A contains a description of the number systems applicable to the Multiprogrammer system. Included are explanations of the decimal, octal, binary, and BCD systems, how to convert numbers from one system to another, and how the number systems relate to some of the plug-in cards.

3-30 Control Mode Reference Table. Table 3-4 lists all of the plug-in cards available at the time of this writing and which operating mode bits (in a control word) will affect the operation of each card. Also included is an IRQ column which indicates the cards that contain IRQ circuitry.

3-31 The following two examples show how to use Table 3-4:

1. If the user has a 69321B D/A card, he can see at a glance that it will be affected by the DTE and SYE control mode bits. He can also determine that if TME is on (extended handshake), it will require 30 usec (nominal) for the card to return a CTF flag.
2. If the user has a 69431A Digital Input card, he can deduce that ISL must be on to read-in data from the card and that the card has interrupt capability (IEN and IRQ). The user can also determine from the table that this card is capable of providing a gate/flag data transfer cycle with its external device when TME is programmed on. In this case, the length of the flag period is, of course determined by the external device (Note 1 of table).

3-32 Mnemonic Tables. The tables that defined the mnemonic codes, Tables 3-1 and 3-2, will be useful in making the program easier to read and understand if used as illustrated in the examples. They are also helpful if the user is interested in generating special programming constants for control or data words to be used in the application program.

3-33 Output Cards, General Programming Techniques

3-34 Three Multiprogrammer control bits (SYE, DTE, TME) are used when programming output cards. Table 3-4 lists the cards affected by these bits and Table 3-1 shows the mnemonics that are used to construct the control words used for output card programming. Explanations of sending constants and variables to output cards as well as descriptions of each control mode (SYE, DTE, TME) are given in the following paragraphs. Programming examples are included where applicable.

NOTE

A unit address of 0 is used in all control words shown in the programming examples. The optional "Unit0" mnemonic has been omitted from the control words to allow easier understanding of the basic programming concepts presented.

3-35 Sending Constants and Variables. As a rule, output cards are programmed with control words and data words. (There are some output cards that can also be operated in the input mode as

will be pointed out in the individual card descriptions in Chapter V.) To send constants, the slot address is added to the data constant that is to be sent. Example 7a shows how to send a constant data value to the card in slot 8. Example 7b shows how to send variable data to the card in slot 5. Notice that the only difference is that a variable, "Voltage", is added to the slot address instead of the constant. In both cases the data must be limited in range from 0 to 4095, otherwise the data portion will overflow into the address portion of the data word causing the wrong card to be programmed. Of course, it is assumed that a control word selecting an output mode and unit address has previously been sent to the Multiprogrammer system.

3-36 The WRITE BIN statement is always used to send either constant or variable data to the Multiprogrammer system. This will send a data to the Multiprogrammer causing it to store the data on the addressed card and start the card's operation (convert the data to a voltage for the 69321B, close the relays for the 69433A, etc.).

3-37 System Enable (SYE). The SYE control mode is used to control the outputs of most output cards (see Table 3-4). To enable the outputs, SYE must be programmed on. When SYE is programmed off, the outputs of all cards affected by SYE are disabled; i.e., are placed in the power on state. For example, resistance outputs are shorted, voltage outputs are 0 volts, and digital outputs are held in the high or low state, depending on the card option.

3-38 Usually a control word with SYE on is programmed before any data words are sent to the Multiprogrammer. In some instances, the user may wish to take advantage of the SYE off condition to preset the state of many output cards so they will all assume an initial state simultaneously when SYE is programmed on. Example 8 shows this parallel initialization for cards in slots 5 and 7. SYE is left on in all subsequent control words.

3-39 Another way the user may wish to take advantage of the SYE off condition is to simultaneously shut down the outputs of all the output cards affected by SYE. This can be accomplished by sending a control word with SYE off to the Multiprogrammer (not including the "Sye" mnemonic in the control word), as shown in Example 9. An important point to remember is that all cards, in all of the mainframes, that have SYE capability will assume their power on state.

3-40 If the user is not using either of the special uses for SYE described above, SYE should be included in all control words sent to the Multiprogrammer. Failure to include SYE will result in its being programmed off causing all cards affected to assume their power on state.

3-41 Data Transfer Enable (DTE). The DTE control mode affects output cards having either external gate/slew capability or dual card storage. DTE need not be programmed on if a particular card does not have either of those capabilities, but no harm is done if it is programmed on. In fact, it is generally recomended that DTE be programmed on in all output control words unless the user needs to use one of the capabilities that will be described later, and then only turn DTE off during those programming

sequences. In addition, when using timing mode (to be described in the next section), DTE and TME should be programmed on together. The reason for this is that the cards affected by DTE cannot drive the CTF flag line unless DTE is turned on. Normally, when DTE is on and one of the cards affected by DTE and TME is addressed and gated, it will immediately return the leading edge of the CTF flag. Depending on the card model, the trailing edge of the flag is either controlled by circuits on the card itself or by external circuits connected to the card. Note that if DTE is not on, the leading edge of the CTF flag cannot be returned, preventing the proper progression of the handshake and the completion of the data transfer from the computer and the Multiprogrammer system.

3-42 Cards With External Gate/Flag Handshake Capability. The Digital Output card (69331A) and the Relay Output card (69330A) have external gate/flag handshake capability. This feature allows these cards to transfer data to an external device which requires a gate input to receive data.

3-43 Normally DTE will be programmed on in a control word, before sending a data word to the card, as previously shown in Example 3. As each data word is received, it is loaded into the addressed card and immediately strobed (gated) into the external device.

3-44 Some multiple card applications may require data to be loaded into several output cards, then simultaneously gated to external devices. In this case, a control word may be sent with DTE off, followed by appropriate data words, then another control word with DTE on. As each data word is received, the data will be immediately transferred to the output of the addressed card. However, the gate from the output card will not be enabled until a control word containing DTE is received.

3-45 Example 10 illustrates using this technique to load data sequentially into two output cards located in slots 1 and 2. Notice that SYE is on, allowing the data to be immediately transferred to the output of the cards, and maintaining the output of other previously programmed output cards. After all the cards have been loaded with data, DTE is turned on, resulting in the simultaneous generation of gate signals from all the programmed cards to their external device.

3-46 Cards With Dual Rank Storage. The D/A voltage (69321B) and Current (69370A) Converter output cards have dual rank storage capability. This feature allows the output of any number of D/A cards to be changed to new values simultaneously. In this case, a control word is sent with DTE off, followed by appropriate data words, then another control word with DTE on. As each data word is received, the data will be loaded into the first rank storage of the addressed card. When a subsequent control word with DTE is received, all previously addressed D/A cards will simultaneously transfer the contents of the first rank storage into the second rank storage and to the output D/A conversion circuits.

3-47 Example 10 also illustrates this technique. In this case, the first rank storage of the D/A cards located in slots 1 and 2 will be loaded with new data. After all the cards have been

loaded with new data, DTE is programmed on, resulting in the simultaneous transfer of data to the second rank storage of all the addressed cards and the conversion to the programmed analog values.

3-48 In either case, if DTE were programmed on before sending data words to the output cards, as each card received its data, it would be loaded into the card and processed immediately.

3-49 Timing Mode or Extended Handshake Mode (TME). The TME control mode allows output cards with CTF circuits to control the Multiprogrammer flag. Table 3-4 lists the cards which may be used in the TME mode. When programming output cards with TME on, the receipt of the trailing edge of the CTF flag will notify the computer that the output card(s) has completed the data transfer (timed out).

3-50 Serial or parallel methods may be used when programming output cards in the timing mode. The serial method programs one output card at a time, waiting for it to complete before programming the next card. The parallel method allows the programming of several output cards at once, and then waits for the last card to complete before allowing further communication with the Multiprogrammer.

3-51 Serial Method. Output cards can be programmed sequentially by first sending a control word with TME and DTE on, and then sending the data words to the cards to be programmed. Each card will take as long as it requires to process the data, returning the trailing edge of the CTF flag when it completes its operation. In this mode of operation the Multiprogrammer automatically paces the output data to the speed required by the external devices or the timing circuits on the output cards.

3-52 Example 11, illustrating this technique, programs output cards located in slots 1, 2 and 5 of the Multiprogrammer. A control word is sent with TME and DTE on (of course SYE is also included to enable the outputs), and then the data words for each of the Multiprogrammer slots that contain an output card that will be programmed. When each card "times out" the data word for the next card is sent.

3-53 This technique is used when the test or process being controlled requires that one output be complete before the next is started for the proper operation of the system.

3-54 Parallel Method. If the test or process being controlled is such that many output cards may be changing at the same time, and it is only required to know when they have all completed, the parllel method should be used. To use the parallel method a control word is sent to the Multiprogrammer with only DTE and SYE on, then the data words for the output cards that will be operating in parallel are sent, followed by a control word with DTE, SYE, and TME on (see Example 12). Notice that the data words will still be sent sequentially to each card, but because TME is not on yet, the computer does not have to wait for the CTF flag. After the control word with TME is programmed, the computer has to wait for the CTF flag to be returned before further transfers can be made to the Multiprogrammer.

3-55 If it is required to know when all the cards have completed their operation, there are two methods to choose from. The simplest method halts the execution of the computer program until all the cards have finished. The second method allows the computer program to continue executing, and notifies the program when all the cards have timed out by an interrupt. This allows the computer to perform calculations, print results, or any other operation that does not involve the Multiprogrammer, until all the cards have completed their operation.

3-56 Stacking Words. The first method is easily programmed by using another WRITE BIN statement with a "dummy" control word. Example 13 shows this technique. When the computer tries to send the dummy control word, it first tests the flag line to see if the CTF flag has been returned. If the flag is still "busy", the computer waits until the flag signals "ready" (trailing edge of CTF returned), before it can send the control word to the Multiprogrammer. During the time the computer is waiting for the flag it cannot be performing other operations, and the dummy control word is said to be "stacked up" waiting to be sent.

3-57 Interrupts. Although the interrupt method is more complex, it allows the computer to perform other operations while it is waiting for the cards to complete. There are four required steps for using interrupts:

1. Establish the linkage. This step tells the computer where it will find instructions on how to handle the interrupt. The statement used is the "ON INT #9 GOSUB Isr". This tells the computer if it receives an interrupt from select code 9 it should branch to a subroutine named "Isr" (for Interrupt Service Routine).
2. Set the conditions that can cause the interface to generate an interrupt. This is done with the "CONTROL MASK 9;128" statement.
3. Enable the interrupts. In this step the interface card (98032A) is told that when the flag signals "ready", it should generate an interrupt. The "CARD ENABLE 9" statement is used for this purpose.
4. Returning from the interrupt service routine. Obviously, this step allows the program to continue where it was interrupted. The statement used for this step is "RETURN". It is placed at the end of the interrupt service routine.

3-58 The interrupt service routine is defined by the user to perform whatever operation is required after all the output cards have completed their operation. It may be as simple as printing "Done" or be a complete test procedure. In the examples given in this guide, the interrupt service routine will usually be very simple, not involving more than a few steps.

3-59 Example 14 illustrates this technique. The operations that will be performed while the computer is waiting for an interrupt are left for the user to define.

3-60 The two techniques described above for determining when the output cards have completed are both called "Waiting for Completion", and can also be used in the serial method of programming in TMC mode if the user needs to know when the las

card has finished. Similar techniques, described in the next section, are also used for input cards.

3-61 Avoiding "Hang-Up" Conditions in TME Mode. During the TME mode, there is a wait-for-flag interval that starts at the time the output card is addressed with a gate and ends when the card returns a CTF flag. During the wait-for-flag period, the computer is said to "hang-up" if another output or input operation is attempted. This condition is used in the stacking words method of waiting for completion described in Paragraph 3-56. There is the possibility that the computer will remain in this state indefinitely, waiting for a CTF that can never be returned, if one of the following conditions exists:

1. The card's external gate/flag circuitry has become defective or disconnected.
2. A card without CTF circuitry (Table 3-4, Note 5) was addressed and gated (with a data word) with TME on.
3. DTE was not turned on with TME and a card affected by DTE (Table 3-4) was addressed and gated.
4. An empty slot or non-existant mainframe was addressed and gated.

3-62 In all of the cases mentioned in the previous paragraph, the fact that the computer "hung-up" notifies the user that there is a problem, either with the hardware in the system, or with the program. For problems 2, 3, and 4 the condition can be cleared by executing a "RESET 9" command, or by placing the 6940B in LOCAL and pressing the RETURN DATA button. Problem (1) can only be solved by toggling the 6940B's LINE switch off and then on. This will cause all the cards and control logic in the 6940B mainframe to assume the power on state, resulting in the need to restore the Multiprogrammer system to the conditions that existed before the problem occurred. It is important to remember that this condition indicates that a problem exists, and this problem should be fixed before the system is restarted. As an aid to determining the where the problem is located, the 6940B's front panel will indicate this condition by the LOAD OUTPUT light remaining constantly on, and the four most significant bits, labeled SLOT ADDRESS, will indicate the card address that is having the problem. It is up to the user, by analyzing the program, to determine which mainframe is presently being addressed (the most recent control word indicates this), and then locate the cause of the problem.

3-63 Input Cards, General Programming Techniques

3-64 Control Mode Bits for Input Card Operations. As indicated in Table 3-4, three control mode bits are applicable to input card programming: ISL, TME, and IEN. The input select (ISL) bit must be programmed on prior to any basic read sequence in which return data is sent back to the computer from an input card. The TME and IEN bits are used to establish the timing and interrupt modes of operation, as will be explained subsequently. Table 3-1 lists the mnemonics that that are used for input card operations.

NOTE

A unit address of 0 is used in all control words shown in the programming examples. The optional "Unit0" mnemonic has been omitted from the control

words to allow easier understanding of the basic programming concepts presented.

3-65 It is important to remember to keep SYE programmed on in all control words sent in the input mode if there are output cards in the system. If SYE is programmed off, the outputs of these cards would be disabled, which may create an undesirable condition. Therefore, all examples shown will include the "SYE" mnemonic in all control words.

3-66 Basic Input Operating Modes. There are two basic programming modes for input cards; serial and parallel. A serial operation can involve the programming of a single input card or a group of cards, each programmed one at a time. A parallel operation involves the simultaneous programming of a group of cards using the IEN mode. In each case a return data word will be read by the computer, as discussed in Paragraphs 3-21 through 3-26, and the data will have to be processed for use in the application program.

3-67 Removing IRQ From the Return Data Word. The return data word was described in Paragraphs 3-21 through 3-26. The return data word may be used directly if it does not contain the IRQ bit, but, if IRQ is present, it must be removed from the data before use by the user's program. Testing for and removing IRQ can easily be accomplished as shown in Example 15. Remember that the IRQ bit is the sign bit in a 16-bit two's complement representation and has a decimal value of -32768. Thus, if the return data word is negative IRQ is present, while a positive value indicates that IRQ was not returned. If it is sure that IRQ will always be returned, as in the case of a 69422A used with TME, and its presence is not of importance, the test can be omitted and just the removal portion retained.

3-68 Serial Operation. There are two programming techniques that are used with input cards in the serial mode. Timing or extended handshake mode (TME on) is used with most of the input cards. This method requires using a "Waiting for Completion" technique to guarantee that the input operation has completed and new data will be read. The address and read without gate can be used with a few "special" I/O cards. It is also used with the IEN mode of input.

3-69 Timing (Extended Handshake) Mode. The TME mode allows input cards with CTF circuits to control the Multiprogrammer flag. Table 3-4 lists the cards which may be used in the TME mode. When programming input cards with TME on, the receipt of the trailing edge of the CTF flag will notify the computer that the input card has completed its' operation (timed out).

3-70 There are two methods of using the timing or extended handshake mode (TME on). The simplest to use involves addressing the input card without a gate and using a READDIN statement to read the return data word into the computer. The READDIN statement causes a gate to be sent to the Multiprogrammer initiating the card's operation and possibly an external gate/flag handshake transfer cycle between the card and its external device. The READBIN will wait for the transfer to complete, and then the new data on the input card is read into the computer. Example 16 demonstrates this method.

3-71 The second method available, using interrupts, although somewhat more complex, has the advantage that the computer can be used to perform other operations, not related to the Multiprogrammer, while waiting for a long input operation to complete. The technique is similar to that used with output card programming and the user is referred to Paragraphs 3-57 and 3-58 for a description of the computer's interrupt requirements. The differences in the programming are illstrated in Example 15. Notice that in this case the address word is sent with a gate and then interrupts are enabled. When the input card operation is complete, the computer will be interrupted and branch to the interrupt service routine labelled "Isr_input". This routine will read the data from the card and store it in variable "Rdw". The READ IO statement is used to read the card without a gate. The "4" in the READ IO statement specifies the primary data register of the interface. The user is referred to "BASIC Language Interfacing Concepts" (09835-90600) for a complete discussion of the interface registers.

3-72 It is important to note that the "with gate" and "without gate" computer operations are always used in pairs for timing mode inputs. That is, if the card is addressed without a gate (WRITE IO) it is always read with a gate (READBIN), and conversely, if it is addressed with a gate (WRITE BIN) it is read without a gate (READ IO). If the card is both addressed and read with a gate, the computer will be required to wait two complete gate/flag handshake cycles and, possibly incorrect operation may result if the external device is such that only one read is allowed per data item.

3-73 Avoiding "Hang-Up" Conditions in TME Mode. During the TME mode, there is a wait-for-flag interval that starts at the time the input card is gated and ends when the card returns a CTF flag. The computer is said to "hang-up" during the wait-for-flag period if another input or output operation is attempted before the flag is returned, or, in the case of the READ BIN statement, while it is waiting for the flag. There is the possibility that the computer will remain in this state indefinitely, waiting for a CTF that can never be returned, if one of the following conditions exists:

1. The external gate/flag circuitry has become defective or disconnected.
2. A card without CTF circuitry (Table 3-4, Note 5) was addressed and gated with TME on.
3. An empty slot or nonexistant mainframe was addressed and gated.

3-74 In all of the cases mentioned above, the fact that the computer "hangs-up" notifies the user that there is a problem, either with the hardware in the system, or with the program. For problems 2 and 3 the condition can be cleared by executing a "RESET 9" command, or by placing the 9540B in LOCAL and pressing the RETURN "+TA button. Problem 1 can only be solved by toggling the 9540B's line switch off and then on. This will cause all the cards and control logic in the 9540B mainframe to assume the power on states, resulting in the need to restore the Multiprogrammer system to the conditions that existed before the problem occurred. It is important to remember that this

condition indicates that a problem exists, and this problem should be fixed before the system is restarted. As an aid in determining where the problem is located, the 6940B's front panel will indicate this condition by the LOAD OUTPUT light remaining constantly on, and the four most significant bits, lablelled SLOT ADDRESS, will indicate the card address that is having the problem. It is up to the user, by analyzing the program, to determine which mainframe is presently being addressed (the most recent control word indicates this), and then locate the cause of the problem.

3-74 Address and Read Without Gate. This method is used with a few of the I/O cards that do not require a gate/flag handshake or for operations with cards using IEN mode. Example 17 shows this technique. First, a control word with ISL and SYE is sent (and gated) to the Multiprogrammer to select input mode. The card to be read is addressed without a gate using the WRITE IO statement, and then the data is immediately read into the computer with the READ IO statement. This is the only recommended way to read data from those I/O cards that do not have CTF circuitry.

3-75 Parallel Mode (IEN). The IEN mode allows multiple input cards to be armed (activated) simultaneously and provides an interrupt to the computer when the first card completes its' operation. Only those input cards listed in Table 3-4 that have IEN capability can be used in the parallel input mode. The cards have two ways of being armed: individually or as a group. The preferred method is to individually arm the cards, as this allows more control over the operation. To enable cards for individual arming, the "Group Arming" jumper W6 must be removed from the card. Group arming can be used if the user is fully aware of all the implications (see Paragraph 7-59).

3-76 Individually Armed Cards. Input cards can be selectively armed (W6 removed) by first sending a control word with ISL on, and then sending an address word with a gate to each card. A control word with IEN and TME is then sent to establish the IEN mode and the computer interrupt system is enabled. The first armed card to complete returns the CTF flag, generating an interrupt in the interface. The interrupt service routine will conduct an IRQ poll to determine which card caused the interrupt. Generally, TME and IEN are turned off during the IRQ poll to prevent further interrupts. Each armed card is then addressed and read without a gate, and the return data word is examined for the presence of IRQ. If IRQ is present it indicates that the addressed card has interrupted and its data is valid. The absence of IRQ signifies the addressed card has not yet generated an interrupt. Note that more than one card may request an interrupt at a single time, thus all cards should be polled.

3-77 After the cards have been polled, the interrupting card(s) must be either recycled or disarmed to prevent them from generating further interrupts from the same conditions. Table 3-5 indicates the requirements for recycling and disarming each card type that can be used in the IEN mode. Additional information can be found in the Operating and Service manuals for each card. To resume IEN mode and wait for further interrupts, a control word with TME and IEN is sent again, and the computer's interrupt system is re-enabled. Example 18 illustrates this method.

3-78 At the end of the IRQ poll, the user may wish to begin a non-IEN operation. In this case, it is recommended to disarm any input cards that have been armed and have not yet interrupted. This will avoid any possibility of erroneous results in other operating modes.

3-79 Another possibility is that at some time after the IEN mode has been established, the user may want to leave the IEN mode and perform some other task with the Multiprogrammer. A possible reason for this is may be an interrupt from a device timeout as determined by using the SET TIMEOUT statement available in the computer. (See the computer's I/O manuals for a discussion of the timeout statements.) To leave the IEN mode the computer's interrupt system must be disbled and the "RESET 9" command must be executed; see Example 19. After the other tasks have been completed, IEN mode can be returned to by sending a control word with TME and IEN on to the Multiprogrammer and re-enabling the computer's interrupt system. Caution is necessary when using this procedure with 69434A Event Sense cards, as these cards do not store interrupts that occur when the Multiprogrammer is not operating in the interrupt mode.

3-80 The differences when using group arming (W6 installed) are in the way card arming is performed. All cards are simultaneously armed when a control word with TME and IEN on is programmed and the individual addressing is not required. Although the group arming saves programming steps, it can be difficult to use unless all input cards in the system are being used in the same manner and the user fully understands all of the implications (read Paragraph 7-59). Therefore, it is recommended that the user remove jumper W6 from all cards having an interrupt capability and use the individual arming method.

CHAPTER IV

HP-IB PROGRAMMING FUNDAMENTALS

4-1 This chapter outlines the fundamentals of programming the Multiprogrammer system with the 9835A or 9845B Desktop Computer using the 98034A HP-IB Interface. Included in this chapter are explanations and examples showing how to program the basic word formats, HP-IB compatability of the Multiprogrammer, as well as general programming techniques for Multiprogrammer input and output cards. All examples assume the HP-IB interface is assigned a select code of 7.

NOTE

4-2 HP-IB INTERFACE FUNCTIONS

4-3 The multiprogrammer system (59500A and 6940B/6941B) is capable of performing the following functions on the HP-IB; Talk, Listen, Request Service and respond to a Serial Poll. The Multiprogrammer cannot act as a controller, extended talker or listener, and will not respond to remote-local,parallel poll, device clear, or device trigger commands.

4-4 MULTIPROGRAMMER WORD FORMATS

4-5 The next portion of this chapter is devoted to the fundamentals of communicating between the Desktop Computer and the Multiprogrammer system. For computer to Multiprogrammer exchanges, three different types of words can be sent, and for Multiprogrammer to computer exchanges two types of words can be received. The following paragraphs describe the basic construction of each word type, showing how to send the output words or read the input words. Each word type is treated separately without too much regard for the programming sequences which can be used to accomplish a specific task. Programming examples, using these basic word formats, are included later in this chapter, in the General Programming Techniques sections.

4-6 Sending Control, Data, and Address Words

4-7 Multiprogrammer Listen Address. Before any of the words can be sent, the Multiprogrammer must first be addressed to listen. Throughout this chapter it is assumed that the address of the Multiprogrammer system has been set to listen and talk addresses of "7" and "W" respectively (see Paragraphs 2-28 through 2-30). This corresponds to an address of "23" as defined in the I/O ROM manuals for the 9835A and 9845B. The BASIC language instruction used to address the Multiprogrammer as well as send the various word types is illustrated in Example 1. The command "OUTPUT" writes to the device whose address is "723", using the "IMAGE" specified by the "USING", the data that is to the right of the ";". The IMAGE statement defines the format the data will be OUTPUT with, the IMAGE used with the Multiprogrammer is "#,K". The "#" suppresses the carriage-return and line-feed characters

normally output and the "K" causes the data to be output as ASCII characters without leading or trailing blanks.

4-8 Control Word. The first word sent to a Multiprogrammer system is a control word. It is used to specify a unit address and the operating mode of the Multiprogrammer system. The unit address portion selects which mainframe in the chain (of one 6940B Multiprogrammer and up to fifteen 6941B Extenders) will respond to subsequent data or address words from the computer. The mode portion of the word selects one of several output or input modes of Multiprogrammer operation (System Enable, Timing Mode Enable, Input Select, etc.). The mode selected affects the entire Multiprogrammer system, including all 6941B Extender units. How the Multiprogrammer system responds to these various control word modes is described in Chapter VII of this guide, as well as in the 6940B Operating and Service Manual. A thorough understanding of these modes is essential for effective programming involving specific applications.

4-9 Figure 4-1 illustrates the structure of a control word as transmitted by the HP-IB interface. The six control word characters are transmitted and in the order written, from left to right. The address character "0", is the control word tag, indicating to the 6940B that the remaining four digits are to be interpreted as a control word, containing operating mode and unit address information.

4-10 The data field consists of four octal digits. The most significant digit, D4, conveys no information in a control word and is always sent as a "0", or not transmitted at all (see note). The operating mode of the Multiprogrammer system is specified in digits D3 and D2. The most commonly used operating mode codes are shown in Table 4-1. Other codes are possible and can be constructed using Figure 4-2.

NOTE

Because of the way the Multiprogrammer system processes the octal characters in the data field, any leading "0"'s need not be sent. Of course, trailing "0"'s must always be transmitted.

4-11 The unit address of the mainframe that subsequent transfers will be made with is contained in digits D2 and D1. The unit address is determined by the physical position of the unit in the chain. The first unit in the chain, the 6940B, has a unit address of 00, the second unit, a 6941B, has an address of 01, and so forth, up to a unit address of 15 (17@8). The two unit address digits are simply added (in octal) to the last two control mode digits shown in Table 4-1 to complete the control word. For example, if the user wants to select unit number one (the first 6941B) and program the DTE, SYE, and TME modes, a code of 0161 would be in the data field. To select unit number 15 (the last unit) and program the same modes, the data field would contain 0177 (0160@8+17@8).

4-12 The last character in a control word must be a "T". The Multiprogrammer system interprets this character as a "gate" which initiates the gate/flag handshake sequence within the 6940B Multiprogrammer and causes it to store the control word. Example

...

2 shows how a typical control word with DTE and SYE would be sent to the Multiprogrammer.

4-13 Data Word. Data words select and program output cards and some functions of a few input cards. The data word selects an individual card within the mainframe previously specified by a control word, and sends a 12-bit value to it.

4-14 Figure 4-3 shows the construction of the data word as sent to the Multiprogrammer by the HP-IB interface. The address character selects which card in the selected mainframe will receive the four digits of data. Table 4-2 shows the ASCII characters for each slot address. (Mainframe slots 400 through 414 will be referred to as slots 0 through 14 in this guide.) The data field consists of four octal digits that can have any octal value between 0000 and 7777 (0 to 4095 decimal). Appendix A contains the number theory on conversion from octal to decimal and decimal-to-octal. The data transfer character is always a "T" for a data word.

4-15 Example 3 illustrates how to send a data word to the card in slot 5. The variable data is first converted to octal using the OCTAL function of the computer and stored in the variable "Dw". Then the control word of the previous example is sent before the data word is transmitted.

4-16 Address Word. An address word selects the input card that is to send data (a return data word) back to the computer. Before an address word can be sent, a control word must have first selected an input operating mode (with ISL on) and the unit address of the mainframe that contains the card.

4-17 An address word contains just two characters (Figure 4-4). The address character specifies the card that is to send data back to the Desktop Computer. The characters are the same as used in the construction of the data word defined in Table 4-3. The data transfer field can be a "T" (read with gate), a "X" (read without gate), or a "Z" (read live data). Which character to use depends on the specific application and the input card type (refer to Chapters V and VII). A "T" always results in the generation of a gate to the addressed input card, and the data from the input card (return data word) will be stored in the 59500A on the trailing edge of the flag from the Multiprogrammer. Thus, the "T" character is recommended as it waits for the completion of the gate/flag handshake to guarantee the accuracy of the data with most input cards.

4-18 An "X" character is sent when it is necessary to store the return data word from the addressed card immediately, without initiating a gate/flag cycle. It is the "X" character itself that stores the return data in the 59500A as soon as it is received from the input card. The "Z" character allows the return data from the addressed card to continuously pass through the 59500A to its output encoders without being stored. When using the "Z" character, the user must keep in mind that the return data may be continuously changing and can therefore be difficult to interpret.

4-19 Example 4 shows a typical address word sent to an input card installed in slot 3. Notice that the preceding control word

has ISL programmed on to select the input mode, and the address word specifies a read with gate sequence.

4-20 Reading Return Data Words and the Status Byte

4-21 Return Data Word. Before the Multiprogrammer can send a return data word, it must be addressed to talk. The BASIC language statement used to address the Multiprogrammer to talk, as well as to read the return data word into the computer is illustrated in Example 5. The command "ENTER" addresses HP-IB device "723" to talk, and reads data from it, storing the data in variable "Rdw". It is assumed that the address word of the previous example has already been sent to the Multiprogrammer.

4-22 The return data word contains information from an input card previously selected by an address word. Each word contains two pieces of information (see Figure 4-4), an input request (IRQ) character and 4 digits of data. A carriage return/line feed is also sent by the 59500A that is used to tell the computer the return data word is complete.

4-23 The IRQ character can be either a "1" or a "0". The main use of IRQ is in the IEN mode of operation when a "1" indicates that the card has generated an interrupt and a "0" indicates the card did not generate an interrupt. However, a "1" also indicates the input card has returned a flag and the data is valid, while a "0" may indicate that a) the card has not returned a flag (it is still busy), or b) the card does not have IRQ capability. Not all input cards have IRQ capability, refer to Table 4-3 for a list of the capabilities of all the plug-in cards.

4-24 The four octal data characters, representing the data from the selected card, can be any octal number between 0 and 7777 (0 to 4095 decimal). Note that the IRQ bit is not part of the data and if present it must be removed from the data prior to use in the application program. Testing for and removing IRQ will be described later in Paragraph 4-65. For some input cards the user will find it convenient to work with the return data in octal, but in most cases the user will want to do calculations with the data. In these cases, the IRQ character must be removed and the data must be converted to decimal using the computer's DECIMAL function.

4-25 Status Byte. The 59500A of the Multiprogrammer system returns an 8-bit status byte to the computer in response to a serial poll. Serial polling is the method used by the computer to determine which HP-IB device has requested service (set SRQ). The Multiprogrammer will request service only in the timing mode (TMC on) upon completion of a Multiprogrammer flag. Both input and output cards can cause SRQ to be set. Chapter VIII describes the specific operating conditions required for a service request to occur (see Paragraph 8-28).

4-26 There are two ways the SRQ line can be monitored, by using the computer's interrupt system or by testing it under program control. In either case, when SRQ has been detected, the method used to determine which HP-IB device requested service is the same, a serial poll is conducted. The serial poll is conducted in any order the user wishes, but it is recommended that the devices that control critical processes be polled first. Bring c

serial poll on the Multiprogrammer system places it in the serial poll mode, addresses it to talk, and reads the status byte into a variable in the computer. The command to perform the serial poll is shown in Example 6. The status byte will have a value of 64 if the Multiprogrammer requested service or a value of 0 if it did not request service. The serial poll operation also clears the SRQ line of the Multiprogrammer so that service requests can be detected from other HP-IB devices. It is recommended that a serial poll be conducted at the beginning of any program to ensure that the SRQ line starts in a clear state.

4-27 GENERAL PROGRAMMING INFORMATION

4-28 The next portion of this chapter describes the general programming sequences associated with the various operating modes and plug-in cards of the Multiprogrammer system. If a detailed description of Multiprogrammer operation is required, see Chapter VII. For specific examples relating to a plug-in card, see Chapter V.

4-29 Number System Theory. Appendix A contains a description of the number systems applicable to the Multiprogrammer system. Included are explanations of the decimal, octal, binary, and BCD systems, how to convert numbers from one system to another, and how the number systems relate to some of the plug-in cards.

4-30 Control Mode Reference Table. Table 4-3 lists all of the plug-in cards available at the time of this writing and which operating mode bits (in a control word) will affect the operation of each card. Also included is an IRQ column which indicates the cards that contain IRQ circuitry.

4-31 The following two examples show how to use Table 4-3:

1. If the user has a 69321B D/A card, he can see at a glance that it will be affected by the DTE and SYE control mode bits. He can also determine that if TME is on (extended handshake), it will require 30 usec (nominal) for the card to return a CTF flag.
2. If the user has a 69431A Digital Input card, he can deduce that ISL must be on to read-in data from the card and that the card has interrupt capability (IEN and IRQ). The user can also determine from the table that this card is capable of providing a gate/flag data transfer cycle with its external device when TME is programmed on. In this case, the length of the flag period is, of course determined by the external device (Note 1 of table).

4-32 Output Cards, General Programming Techniques

4-33 Three Multiprogrammer control bits (SYE, DTE, TME) are used when programming output cards. Table 4-3 lists the cards affected by these bits and Table 4-1 shows the control codes most often used for output card programming. Explanations of sending constants and variables to output cards as well as descriptions of each control mode (SYE, DTE, TME) are given in the following paragraphs. Programming examples are included where applicable.

NOTE
A unit address of 00 is used in all control

words shown in the programming examples.

4-34 Sending Constants and Variables. As a rule, output cards are programmed with control words and data words. (There are some output cards that can also be operated in the input mode as will be pointed out in the individual card descriptions in Chapter V.) The data field portion of the data word may contain a constant (explicit value) or a variable. In either case, the data is sent using the OUTPUT statement described in Paragraph 4-7. The constant in the data field can be any octal value in the range of 0000 to 7777. Remember that when using variable data it must be converted to octal (using the OCTAL function), and must be limited to a range of 0000 to 7777 octal (0 to 4095 decimal). Of course if the data is already octal, the OCTAL function will not be used. Example 7 shows a constant data word and a variable data word being ouput to the Multiprogrammer.

4-35 System Enable (SYE). The SYE control mode is used to control the outputs of most output cards (see Table 3-4). To enable the outputs, SYE must be programmed on. When SYE is programmed off, the outputs of all cards affected by SYE are disabled; i.e., are placed in the power on state. For example, resistance outputs are shorted, voltage outputs are 0 volts, and digital outputs are held in the high or low state, depending on the card option.

4-36 Usually a control word with SYE on is programmed before any data words are sent to the Multiprogrammer. In some instances, the user may wish to take advantage of the SYE off condition to preset the state of many output cards so they will all assume an initial state simultaneously when SYE is programmed on. Example 8 shows this parallel initialization for cards in slots 5 and 7. SYE is left on in all subsequent control words.

4-37 Another way the user may wish to take advantage of the SYE off condition is to simultaneously shut down the outputs of all the output cards affected by SYE. This can be accomplished by sending a control word with SYE off to the Multiprogrammer, as shown in Example 9. Since leading zeroes need not be sent, the control word contains only the control word tag "0" and the "T" to gate the Multiprogrammer. An important point to remember is that all cards, in all of the mainframes, that have SYE capability will assume their power on state.

4-38 If the user is not using either of the special uses for SYE described above, SYE should be included in all control words sent to the Multiprogrammer. Failure to include SYE will result in it being programmed off causing all cards affected to assume their power on state.

4-39 Data Transfer Enable (DTE). The DTE control mode affects output cards having either external gate/flag capability or dual rank storage. DTE need not be programmed on if a particular card does not have either of these capabilities, but no harm is done if it is programmed on. In fact, it is generally recommended that DTE be programmed on in all output control words unless the user needs to use one of the capabilities that will be described later, and then only turn DTE off during those programming sequences. In addition, when using timing mode (to be described in the next section), DTE and TME should be programmed on

together. The reason for this is that the cards affected by DTE cannot drive the CTF flag line unless DTE is turned on. Normally, when DTE is on and one of the affected is addressed and gated it will immediately return the leading edge of the CTF flag. Depending on the card model, the trailing edge of the flag is either controlled by circuits on the card itself or by external circuits connected to the card. Note that if DTE is not on, the leading edge of the CTF flag cannot be returned, preventing the proper progression of the handshake and the completion of the data transfer from the computer and the Mutiprogrammer system.

4-40 Cards With External Gate/Flag Handshake Capability. The Digital Output card (69331A) and the Relay Output card (69330A) have external gate/flag handshake capability. This feature allows these cards to transfer data to an external device which requires a gate input to receive data.

4-41 Normally DTE will be programmed on in a control word, before sending a data word to the card, as previously shown in Example 3. As each data word is received, it is loaded into the addressed card and immediately strobed (gated) into the external device.

4-42 Some multiple card applications may require data to be loaded into several output cards, then simultaneously gated to external devices. In this case, a control word may be sent with DTE off, followed by appropriate data words, then another control word with DTE on. As each data word is received, the data will be immediately transferred to the output of the addressed card. However, the gate from the output card will not be enabled until a control word containing DTE is received.

4-43 Example 10 illustrates using this technique to load data sequentially into two output cards located in slots 1 and 2. Notice that SYE is on, allowing the data to be immediately transferred to the output of the cards, and maintaining the output of other previously programmed output cards. After all the cards have been loaded with data, DTE is turned on, resulting in the simultaneous generation of gate signals from all the programmed cards to their external device.

4-44 Cards With Dual Rank Storage. The D/A voltage (69321D) and Current (69370A) Converter output cards have dual rank storage capability. This feature allows the output of any number of D/A cards to be changed to new values simultaneously. In this case, a control word is sent with DTE off, followed by appropriate data words, then another control word with DTE on. As each data word is received, the data will be loaded into the first rank storage of the addressed card. When a subsequent control word with DTE is received, all previously addressed D/A cards will simultaneously transfer the contents of the first rank storage into the second rank storage and to the output D/A conversion circuits.

4-45 Example 10 also illustrates this technique. In this case, the first rank storage of the D/A cards located in slots 1 and 2 will be loaded with new data. After all the cards have been loaded with new data, DTE is programmed on, resulting in the simultaneous transfer of data to the second rank storage of all the addressed cards and the conversion to the programmed analog

values.

4-46 In either case, if DTE were programmed on before sending data words to the output cards, as each card received its data, it would be loaded into the card and processed immediately.

4-47 Timing Mode or Extended Handshake Mode (TME). The TME control mode allows output cards with CTF circuits to control the Multiprogrammer flag. Table 4-3 lists the cards which may be used in the TME mode. When programming output cards with TME on, the receipt of the trailing edge of the CTF flag will notify the computer that the output card(s) has completed the data transfer (timed out).

4-48 Serial or parallel methods may be used when programming output cards in the timing mode. The serial method programs one output card at a time, waiting for it to complete before programming the next card. The parallel method allows the programming of several output cards at once, and then waits for the last card to complete before allowing further communication with the Multiprogrammer.

4-49 Serial Method. Output cards can be programmed sequentially by first sending a control word with TME and DTE on, and then sending the data words to the cards to be programmed. Each card will take as long as it requires to process the data, returning the trailing edge of the CTF flag when it completes its operation. In this mode of operation the Multiprogrammer automatically paces the output data to the speed required by the external devices or the timing circuits on the output cards.

4-50 Example 11, illustrating this technique, programs output cards located in slots 1, 2 and 5 of the Multiprogrammer. A control word is sent with TME and DTE on (of course SYE is also included to enable the outputs), and then the data words for each of the Multiprogrammer slots that contain an output card that will be programmed. When each card "times out" the data word for the next card is sent.

4-51 This technique is used when the test or process being controlled requires that one output be complete before the next is started for the proper operation of the system.

4-52 Parallel Method. If the test or process being controlled is such that many output cards may be changing at the same time, and it is only required to know when they have all completed, the parllel method should be used. To use the parallel method a control word is sent to the Multiprogrammer with only DTE and SYE on, then the data words for the output cards that will be operating in parallel are sent, followed by a control word with DTE, SYE, and TME on (see Example 12). Notice that the data words will still be sent sequentially to each card, but because TME is not on yet, the computer does not have to wait for the CTF flag. After the control word with TME is programmed, the computer has to wait for the CTF flag to be returned before further transfers can be made to the Multiprogrammer.

4-53 If it is required to know when all the cards have completed their operation, there are two methods to choose from. The simplest method halts the execution of the computer program until

all the cards have finished. The second method allows the computer program to continue executing, and notifies the program when all the cards have timed out by an interrupt. This allows the computer to perform calculations, print results, or any other operation that does not involve the Multiprogrammer, until all the cards have completed their operation.

4-54 Stacking Words. The first method is easily programmed by outputing "dummy" control word. Example 13 shows this technique. When the computer tries to send the dummy control word, it first tests the flag line to see if the CTF flag has been returned. If the flag is still "busy", the computer waits until the flag signals "ready" (trailing edge of CTF returned), before it can send the control word to the Multiprogrammer. During the time the computer is waiting for the flag it cannot be performing other operations, and the dummy control word is said to be "stacked up" waiting to be sent.

4-55 Interrupts. Although the interrupt method is more complex, it allows the computer to perform other operations while it is waiting for the cards to complete. There are four required steps for using interrupts:

1. Establish the linkage. This step tells the computer where it will find instructions on how to handle the interrupt. The statement used is the "ON INT #7 GOSUB Isr". This tells the computer if it receives an interrupt from select code 7 it should branch to a subroutine named "Isr" (for Interrupt Service Routine).
2. Set the conditions that can cause the interface to generate an interrupt. This is done with the "CONTROL MASK 7;128" statement.
3. Enable the interrupts. In this step the interface card (98034A) that when the SRQ signal is received, it should generate an interrupt. The "CARD ENABLE 7" statement is used for this purpose.
4. Returning from the interrupt service routine. Obviously, this step allows the program to continue where it was interrupted. The statement used for this step is "RETURN". It is placed at the end of the interrupt service routine.

4-56 The interrupt service routine is defined by the user to perform whatever operation is required after all the output cards have completed their operation. It may be as simple as printing "Done" or be a complete test procedure. In the examples given in this guide, the interrupt service routine will usually be very simple, not involving more than a few steps.

4-57 Example 14 illustrates this technique. The operations that will be performed while the computer is waiting for an interrupt are left for the user to define.

4-58 The two techniques described above for determining when the output cards have completed are both called "Waiting for Completion", and can also be used in the serial method of programming in TNE mode if the user needs to know when the last card has finished. Similar techniques, described in the next section, are also used for input cards.

4-59 Avoiding "Hang-Up" Conditions in TME Mode. During the TME mode, there is a wait-for-flag interval that starts at the time the output card is gated ("T"), and ends when the card returns a CTF flag. During the wait-for-flag period, the computer is said to "hang-up" if another output or input operation is attempted. This condition is used in the stacking words method of waiting for completion described in Paragraph 4-54. There is the possibility that the computer will remain in this state indefinitely, waiting for a CTF that can never be returned, if one of the following conditions exists:

1. The card's external gate/flag circuitry has become defective or disconnected.
2. A card without CTF circuitry (Table 4-3, Note 5) was addressed and gated (with a data word) with TME on.
3. DTE was not turned on with TME and a card affected by DTE (Table 4-3) was addressed and gated.
4. An empty slot or non-existant mainframe was addressed and gated.

4-60 In all of the cases mentioned in the previous paragraph, the fact that the computer "hung-up" notifies the user that there is a problem, either with the hardware in the system, or with the program. For problems 2, 3, and 4 the condition can be cleared by sending an "X" to the Multiprogrammer or by placing the 6940B in LOCAL and pressing the RETURN DATA button. Problem (1) can only be solved by toggling the 6940B's LINE switch off and then on. This will cause all the cards and control logic in the 6940B mainframe to assume the power on state, resulting in the need to restore the Multiprogrammer system to the conditions that existed before the problem occurred. It is important to remember that this condition indicates that a problem exists, and this problem should be fixed before the system is restarted. As an aid to determining the where the problem is located, the 6940B's front panel will indicate this condition by the LOAD OUTPUT light remaining constantly on, and the four most significant bits, labeled SLOT ADDRESS, will indicate the card address that is having the problem. It is up to the user, by analyzing the program, to determine which mainframe is presently being addressed (the most recent control word indicates this), and then locate the cause of the problem.

4-61 Input Cards, General Programming Techniques

4-62 Control Mode Bits for Input Card Operations. As indicated in Table 4-3, three control mode bits are applicable to input card programming: ISL, TME, and IEN. The input select (ISL) bit must be programmed on prior to any basic read sequence in which return data is sent back to the computer from an input card. The TME and IEN bits are used to establish the timing and interrupt modes of operation, as will be explained subsequently. Table 4-1 lists the most often used control mode codes for input card operations.

NOTE
A unit address of 0 0 is used in all control words shown in the programming examples.

4-63 It is important to remember to keep SYE programmed on in all control words sent in the input mode if cards are output

cards in the system. If SYE is programmed off, the outputs of these cards would be disabled, which may create an undesirable condition. Therefore, all examples shown will include the SYE in all control words.

4-64 Basic Input Operating Modes. There are two basic programming modes for input cards; serial and parallel. A serial operation can involve the programming of a single input card or a group of cards, each programmed one at a time. A parallel operation involves the simultaneous programming of a group of cards using the IEN mode. In each case a return data word will be read by the computer and the data will have to be processed for use in the application program.

4-65 Removing IRQ From the Return Data Word. The return data word was described in Paragraphs 4-20 through 4-24. The return data word may be used directly if it does not contain the IRQ character, but, if IRQ is present, it must be removed from the data before use by the user's program. Testing for and removing IRQ can easily be accomplished as shown in Example 15. If it is sure that IRQ will always be returned, as in the case of a 69433A used with TME, and its presence is not of importance, the test can be omitted and just the removal portion retained.

4-66 Serial Operation. There are two programming techniques that are used with input cards in the serial mode. Timing or extended handshake mode (TME on) is used with most of the input cards. This method requires using a "Waiting for Completion" technique to guarantee that the input operation has completed and new data will be read. The address and read without gate can be used with a few "special" I/O cards. It is also used with the IEN mode of input.

4-67 Timing (Extended Handshake) Mode. The TME mode allows input cards with CTF circuits to control the Multiprogrammer flag. Table 4-3 lists the cards which may be used in the TME mode. When programming input cards with TME on, the receipt of the trailing edge of the CTF flag will notify the computer that the input card has completed its' operation (timed out).

4-68 The first technique uses an additional "dummy" address word to "stack words" similar to the technique used for output. The 59500A cannot be addressed to talk until all the words being sent to it are processed, and with TME on the "dummy" address word is stacked waiting for the input card to complete its operation. This is shown in Example 16.

4-69 The second method available, using interrupts, although somewhat more complex, has the advantage that the computer can be used to perform other operations, not related to the Multiprogrammer, while waiting for a long input operation to complete. The technique is similar to that used with output card programming and the user is referred to Paragraphs 4-55 and 4-56 for a description of the computer's interrupt requirements. The differences in the programming are illstrated in Example 17. Notice that in this case the address word is sent with a gate and then interrupts are enabled. When the input card operation is complete, the computer will be interrupted and branch to the interrupt service routine labelled "Isr_input". This routine will read the data from the card and store it in variable "Data".

4-70 Avoiding "Hang-Up" Conditions in TME Mode. During the TME mode, there is a wait-for-flag interval that starts from at time the input card is gated ("T") and ends when the card returns a CTF flag. The computer is said to "hang-up" during the wait-for-flag period if another input or output operation is attempted before the flag is returned. There is the possibility that the computer will remain in this state indefinitely, waiting for a CTF that can never be returned, if one of the following conditions exists:

1. The external gate/flag circuitry has become defective or disconnected.
2. A card without CTF circuitry (Table 4-3, Note 5) was addressed and gated with TME on.
3. An empty slot or non-existant mainframe was addressed and gated.

4-71 In all of the cases mentioned above, the fact that the computer "hung-up" notifies the user that there is a problem, either with the hardware in the system, or with the program. For problems 2 and 3 this condition can be cleared by sending a "X" to the Multiprogrammer system, or by placing the 6940B in LOCAL and pressing the RETURN DATA button. Problem 1 can only be solved by toggling the 6940B's line switch off and then on. This will cause all the cards and control logic in the 6940B mainframe to assume the power on state, resulting in the need to restore the Multiprogrammer system to the conditions that existed before the problem occurred. It is important to remember that this condition indicates that a problem exists, and this problem should be fixed before the system is restarted. As an aid in determining where the problem is located, the 6940B's front panel will indicate the hung-up condition by the LOAD OUTPUT light remaining constantly on, and the four most significant bits, labelled SLOT ADDRESS, will indicate the card address that is having the problem. It is up to the user, by analyzing the program, to determine which mainframe is presently being addressed (the most recent control word indicates this), and then locate the cause of the problem.

4-72 Address and Read Without Gate. This method is used with a few of the I/O cards that do not require a gate/flag handshake or for operations with cards using IEN mode. Example 18 shows this technique. First, the control word with ISL and SYE is sent (unmasked) to the Multiprogrammer to select input mode. The card to be read is addressed without a gate by using a "X" character in the data transfer field instead of the "T". Then the data is immediately read into the computer with the ENTER statement. This is the only recommended way to read data from those I/O cards that do not have CTF circuitry.

4-73 Parallel Mode (IEN). The IEN mode allows multiple input cards to be armed (activated) simultaneously and provides an interrupt to the computer when the first card completes its operation. Only those input cards listed in Table 4-3 that have IEN capability can be used in the parallel input mode. The cards have two ways of being armed: individually or as a group. The preferred method is to individually arm the cards, as this allows more control over the operation. To enable cards for individual arming, the "Group Arming" jumper W6 must be removed from the

card. Group arming can be used if the user is fully aware of all the implications (see Paragraph 7-59).

4-74 Individually Armed Cards. Input cards can be selectively armed (W6 removed) by first sending a control word with ISL on, and then sending an address word with a gate to each card. A control word with IEN and TME is then sent to establish the IEN mode and the computer interrupt system is enabled. The first armed card to complete returns the CTF flag, generating an interrupt in the interface. The interrupt service routine will conduct an IRQ poll to determine which card caused the interrupt. Generally, TME and IEN are turned off during the IRQ poll to prevent further interrupts. Each armed card is then addressed and read without a gate, and the return data word is examined for the presence of IRQ. If IRQ is present it indicates that the addressed card has interrupted and its data is valid. The absence of IRQ signifies the addressed card has not yet generated an interrupt. Note that more than one card may request an interrupt at a single time, thus all cards should be polled.

4-75 After the cards have been polled, the interrupting card(s) must be either recycled or disarmed to prevent them from generating further interrupts from the same conditions. Table 4-4 indicates the requirements for recycling and disarming each card type that can be used in the IEN mode. Additional information can be found in the Operating and Service manuals for each card. To resume IEN mode and wait for further interrupts, a control word with TME and IEN is sent again, and the computer's interrupt system is re-enabled. Example 19 illustrates this technique.

4-76 At the end of the IRQ poll, the user may wish to begin a non-IEN operation. In this case, it is recommended to disarm any input cards that have been armed and have not yet interrupted. This will avoid any possibility of hang-ups or erroneous results in other operating modes.

4-77 Another possibility is that at some time after the IEN mode has been established, the user may want to leave the IEN mode and perform some other task with the Multiprogrammer. To leave the IEN mode the computer's interrupt system must be disbled and an "X" must be sent to the Multiprogrammer. After the other tasks have been completed, IEN mode can be returned to by sending a control word with TME and IEN on to the Multiprogrammer and re-enabling the computer's interrupt system. Example 20 shows how this is accomplished. Caution is necessary when using this procedure with 69434A Event Sense cards, as these cards do not store interrupts that occur when the Multiprogrammer is not operating in the interrupt mode.

4-78 The differences when using group arming (W6 installed) are in the way card arming is performed. All cards are simultaneously armed when a control word with TME and IEN on is programmed and the individual addressing is not required. Although the group arming saves programming steps, it can be difficult to use unless all input cards in the system are being used in the same manner and the user fully understands all of the implications (read Paragraph 7-59). Therefore, it is recommended that the user remove jumper W6 from all cards having an interrupt capability and use the individual arming method.

CHAPTER V

PLUG-IN CARD DESCRIPTIONS AND PROGRAMS

5-1 This chapter provides 16-bit and HP-IB interface programming examples for each type of plug-in card that can be presently used in a Multiprogrammer system. The cards are presented in numerical sequence according to card number (69321B through 69601B). Block diagrams and input/output connector diagrams are provided as programming and application aids. In addition, several programming examples of typical measurement applications, involving multiple card systems, are provided at the end of the chapter. All examples assume the card is installed in unit 0, the 6940B, the control words will have to be modified to include the unit address if the card is installed in another unit. The I/O cards can be installed in any of the 400 series slots in the mainframe (400-414), throughout this manual the 400 series slot addresses will be referred to as slots 0 through 14. Any subroutines that may be used in the sample programs can be found in Appendix A.

5-2 INDIVIDUAL CARD PROGRAMS

5-3 D/A Voltage Converter Card, 69321B

5-4 The 69321B D/A Voltage Converter card is a twelve bit bipolar (two's complement) digital to analog converter which may be programmed to output a voltage in the range of -10.240 through 10.235 volts. An explanation of the relationship between a voltage and the corresponding twelve bit code is given in Appendix A. A 69351B Voltage Regulator card must be installed in slot 600 of the unit in which the D/A is located to provide the +/- 15 volt bias voltage required by the 69321B.

5-5 Calculating Data Values. The following steps will yield the data value required to program the 69321B to a desired output voltage.

1. Divide the desired output voltage by .005 volts (the LSB is 5 mV.) This determines the number of LSB's in the output voltage.
2. If the result of step 1 is negative, add 4096 to the result. This will convert the data to a twelve bit two's complement value.
3. HP-IB only. Convert the data to octal.

5-6 Programming the 69321B. Example 1 shows how to program a 69321B, located in slot 5, to a voltage selected by the user. A test is included to determine if the desired voltage is within the range of the card. The output voltage will not appear at the terminals unless SYE is programmed on. INT allows multiple cards to be programmed to their respective output values simultaneously by using the dual ramp storage capability of the card. A CTF flag is returned, or TMR may be used. Detailed information on the various operating modes can be found in the Programming Fundamentals chapters.

5-7 Relay Output Card, 69330B

5-8 The 69730A Relay Output card provides 12, independent, SPST normally open (Form A) mercury wetted relays, with 2 additional

relays for a gate/flag function.

NOTE

> It is recommended that this card only be used if the gate/flag function is required by the user's system. If the gate/flag function is not required, the 69433A Realy Readback card is recommended. If the 69330A is used in an application that does not require gate/flag, the gate must be jumpered to the flag to prevent system hang-ups.

5-9 Data Values. The data sent to the card will be in the form of contact closures, a "1" specifying a closed relay and a "0" specifies an open relay in the equivalent 12-bit binary word. Each time a data word is sent to the 69330A, the state of all relays is reprogrammed to conform to the new data. Thus, if only one relay is to be changed, the state of all the relays must be specified; it is up to the user to save this information in the program.

5-10 Programming the 69330A. Example 2 illustrates a typical programming sequence, for a 69330A installed in slot 2, including a test to insure the data is within the correct range. SYE must be programmed on for the relays to assume their programmed state. DTE can be used to generate the gate signal from the Relay Output card. TME is used if the user's program requires notification when then flag from the external device has been returned, note that DTE must be programmed on when using TME. The Programming Fundamentals chapters contain detailed information on how to use the various operating modes.

5-11 Digital Output Card, 69331A

5-12 The 69331A Digital Output card provides twelve 5 volt (TTL/DTL compatiable) or 12 volt logic levels at its output. A gate/flag feature is also provided that can be used to pace the transfer of data to external devices and ensure accurate transfers.

NOTE

> If the gate/flag function will not be used, the gate out pin should be connected to the flag return pin, this will prevent a system hang-up condition.

5-13 Data Values. The data sent to the 69331A must be in the range of 0 to 4095 (0000 to 7777 octal), the corresponding binary bit pattern will appear at the output. A "1" in the equivalent binary representation will cause the corresponding output to be true, a "0" will program the output to a false state. The actual voltage levels depend on the options selected.

5-14 Programming the 69331A. A typical program sequence, for a Digital Output card installed in slot 1, is illustrated in Example 3. The data is checked for correct range before being sent to the card. If SYE is off, it will hold the digital outputs in the logical "0" state. The gate signal will be generated with DTE on, and if TME is on, the Multiprogrammer will wait for the flag to be returned from the external device, thus notifying the

user's program when the transfer is complete.

5-15 Open Collector Output Card, 69332A

5-16 The 69332A Open Collector Output card provides 12 open collector outputs that can be used for driving lamps or relay coils utilizing an external DC power supply.

NOTE

The 69332A is not affected by SYE, therefore its use is recommended only for non-critical outputs that do not need to be quickly shut down.

5-17 Data Values. The data sent to the 69332A must be in the range of 0 to 4095 (0000 to 7777 octal); the corresponding binary bit pattern will appear at the output. A "1" in the equivalent binary representation will cause the corresponding output to be true; a "0" will program the output to a false state.

5-18 Programming the 69332A. A typical programming sequence for an Open Collector Output card, installed in slot 2, is illustrated in Example 4. The data is first checked for proper range before being sent to the card. Although SYE has no effect on the 69332A, it is included in the control word so as not to disturb the outputs of other cards in the system that are affected by SYE.

5-19 Stepping Motor Control Card, 69335A

5-20 The 69335A Stepping Motor Control card can be programmed to generate from 1 to 2047 squarewave pulses at either of two output terminals. When applied to a stepping motor translator, these pluses are converted to clockwise or counterclockwise drive pulses for a stepping motor. The squarewave outputs of the 69335A can also be used for pulse train update of setpoint control stations.

5-21 Data Values. The data sent to the 69335A contains information concerning the number of pulses to be produced and which terminal they should be output from. If Output A (terminal 1) is considered positive and Output B (terminal 2) is considered negative, the data can be thought of as being in a sign/magnitude representation (see Appendix A). The range of data for a 12-bit sign/magnitude word is -2047 to 2047, with the MSB of the 12-bit word being the sign bit. The weight of the MSB is 2048 (4000 octal); thus, if a negative output is desired this value must be added to the absolute value of the number of pulses to select the correct output terminal.

5-22 Programming the 69335A. Example 5 shows a typical programming sequence for a 69335A installed in slot 7. The number of pulses selected by the user is tested for correct range and then converted into the correct format to be sent to the card. The control word contains SYE and TME, the SYE is required to allow the pulses to be output. TME will cause the Multiprogrammer to wait for all pulses to be output before proceeding with the next operation. The Programming Fundamentals chapters contain detailed information on the various operating modes.

5-23 D/A Current Converter Card, 69370A

5-24 The 69370A D/A Current Converter card provides a high speed, constant current output that can be programmed in the range of 0 to 20.475 mA with a resolution of 5 uA. A 69351B Voltage Regulator card is installed in slot 600 of the unit in which the 69370A is located.

5-25 Calculating Data Values. The data value required to program the 69370A to a desired output current can be determined as follows.

1. Divide the desired output current, in milliamps, by .005 mA (the LSB is .005 mA). This determines the number of LSB's in the output current and thus, the program data.
2. HP-IB only. Convert the data to octal.

5-26 Programming the 69370A. Example 6 illustrates a typical program sequence for a 69370A in slot 5. The current entered by the user is first checked to determine if it is in the range of the D/A Current Converter, and then it is converted and sent to the card. The output current will not appear at the output unless SYE is on. DTE and TME are also included in the control word as they both affect the operation of the 69370A. See the Programming Fundamentals chapters for detailed information on how the various control mode may be used.

5-27 Breadboard Output Card, 69380A

5-28 The 69380A Breadboard Output card provides a means of interfacing custom circuits to the Multiprogrammer. It contains 12-bits of storage, logic to accept the 12-bits of data from the Multiprogrammer, and an area for the custom circuits to be constructed. The 69380A also provides means of using the SYE, DTE, and TME control modes.

5-29 Data Values. The data sent to the 69380A must be in the range of 0 to 4095 (0000 to 7777 octal), the corrsponding binary bit pattern will appear at the output. A "1" in the equivalent binary representation will cause the corresponding output to be true, a "0" will program the output to a false state.

5-30 Programming the 69380A. A typical programming sequence, for a Breadboard Output card installed in slot 2, is illustrated in Example 7. The data is checked for correct range before being sent to the card. Although SYE is not effective for the 69380A, it is included in the control word so as not to disturb the outputs of other cards in the system that are affected by SYE.

5-31 High Speed A/D Card, 69422A

5-32 The 69422A High Speed A/D card is a twelve bit, bipolar (two's complement) analog to digital converter capable of measuring DC voltages in one of four user selectable ranges, +/-100, +/-10, +/-1, or +/-0.1 volts. An external trigger is provided to allow synchronizing the measurement with an external event. The 69422A is also capable of interrupting the computer, using the IFC mode, after it has converted a voltage on command from an external trigger. A 69351B Voltage Regulator card must

be installed in slot 600 of the unit in which the A/D is located to provide the +/-15 volt bias voltages required by the 69422A.

5-33 Data Values. The 69422A returns a 12-bit two's complement word which represents the input voltage. An explanation of the relationship between the input voltage and the data is given in Appendix A. As shipped from the factory, the card is set for the +/-10 volt range with an LSB of 5 mV. To convert the data to a voltage the following steps can be used.

1. Remove the IRQ bit. This is explained in Paragraphs 3-67 (for the 16-bit interface) or 4-## (for the HP-IB interface).
2. HP-IB only. Convert the data from octal to decimal.
3. If the data is greater than 2047, subtract 4096 from it.
4. Multiply the result from step 3 by the value of the LSB.

5-34 Programming the 69422A. Example 8 illustrates a typical programming sequence for a High Speed A/D card installed in slot 6. TME is used in the control word to allow the 69422A to convert the voltage to digital data before it is read by the computer. To prevent the outputs of other cards from being disturbed while taking a measurement, SYE is also included in the control word. Detailed information on the various control modes can be found in the Programming Fundamentals chapters. After the data is read by the computer, it is converted to the equivalent voltage and displayed.

5-35 Isolated Digital Input Card, 69430A

5-36 The 69430A Isolated Digital Input card provides a means of eliminating ground loop problems in automatic test and control systems when digital data is required. Each of the 12 inputs is independently photo-isolated from the Multiprogrammer and computer AC earth ground. The proper input voltage option must be ordered as the inputs must be capable of driving the LED's in the photo-isolators. Positive true and negative true input options are also available. The external device must be capable of sourcing 3mA.

5-37 Data Values. The data returned by the Isolated Digital Input card is the range of 0 to 4095 (0000 to 7777 octal). The interpretation of the data, unsigned binary, sign magnitude, two's complement, etc. is dependent on the device connected to the 69430A. Thus, in the example program the data will be treated as an unsigned number.

5-38 Programming the 69430A. A typical program sequence, for a 69430A installed in slot 4, is shown in Example 9. The 69430A cannot use IEN and TME modes, thus only ISL and SYE are included in the control word. The SYE is present only so as not to disturb the outputs of cards that are affected by SYE. The data is displayed after it has been read by th computer.

5-39 Digital Input Card, 69431A

5-40 The 69431A Digital Input card provides a means of reading 12-bits of logic level or contact closure data with an external gate/flag synchronizing handshake. The external handshake can also be used to generate an interrupt to the computer when in the

IEN mode.

NOTE
If the gate/flag function will not be used, the gate out pin must be connected to the flag return pin. Failure to do so may result in an indefintely long wait for a flag that can never be returned.

5-41 Data Values. The data returned by the Digital Input card can be in the range of 1 to 4095 (0000 to 7777 octal). The voltage levels that define the "1" state and the "0" state are determined by the option selected when ordered. The data may have any interpretation assigned by the user. For the purposes of the example program unsigned binary will be assumed.

5-42 Programming the 69431A. Example 10 illustrates a typical programming sequence for a 69431A installed in slot 3. TME is used in the control word to allow the Multiprogrammer to wait for the gate/flag handshake to complete before sending the data to the computer. The control word also contains SYE so the outputs of cards that are affected by SYE will not be disturbed. The Programming Fundamentals chapters contain more information on the use of the various operating modes.

5-43 Relay Output With Readback Card, 69433A

5-44 The 69433A Relay Output with Readback card provides 12, independent, SPST normally open (Form A) mecury wetted relays. In addition, the 69433A can send a data word to the computer that contains the present programmed status of the relays. This allows the card to be treated as individually programmable switches, without requiring the user to remember the present state of the relays before programming a new state. This feature is especially useful when scanning analog signals.

5-45 Data Values. The data sent to the card will be in the form of contact closures, a "1" specifying a closed relay and a "0" specifies an open relay in th equivalent 12-bit binary word. Each time a data word is sent to the 69433A, the state of all relays is reprogrammed to conform to the new data. Thus, if only one relay is to be changed, the status of the presently programmed relays can be read into the computer, the new data combined with the existing data, and the result sent back to the card.

5-46 Programming the 69433A. Example 10 shows a typical program sequence for the Relay Output with Readback card, installed in slot 2, used as individually settable switches. The relay to be closed or opened is entered into the program, the present relay status is read into the computer and combined with the new data, and the result is then programmed back to the 69433A. TME is used in the output control word to ensure the relays have closed before another operation is attempted. See the Programming Fundamentals chapters for additional information on the various operating modes.

5-47 Event Sense Card, 69434A

5-48 The 69434A Event Sense card monitors up to 12 external contact closures and can generate an interrupt to the computer

when one or more contacts change for 20 milliseconds or longer with respect to 12 reference bits stored on the card. The condition that generates the interrupt is selected by jumper W3 as follows:

W3 Position	Interrupting Condition
A	Input Word = Reference Word
B	Input Word > Reference Word
C	Input Word < Reference Word
D*	Input Word # Reference Word

* The card is shipped from the factory with the W3 jumper in the D position.

5-49 The Event Sense card continually compares the input word with a reference word while the Multiprogrammer system is in the IEN mode and stores the entire input word at the moment the event (interrupt) occurs. Both the interrupting input word and the reference word can be read back, after the event has occurred. As shipped from the factory, the 69434A will not store input data unless the IEN mode is programmed on and the gate has been set.

5-50 Data Values. The data returned by the 69434A, whether it is the interrupting input word or the reference word, will be in the range of 0 to 4095 (0000 to 7777 octal). The data has no special meaning assigned to it by the Multiprogrammer (two's complement, sign magnitude, etc.) thus, it is up to the user to define the significance of the return data word. In many cases the data will be treated as individual bits, each with a meaning of their own.

5-51 Programming the 69434A. A typical programming sequence for an Event Sense card, installed in slot 4, is shown in Example 11. This program includes the four basic steps required to program the 69434A.

1. Loading the reference word to initialize the card. The reference word is the 12-bit pattern that the input data is compared to in the determination of an interrupt. It is sent to the 69434A in the output mode (SYE on).
2. Arming the card. This step allows the card to generate an interrupt if the conditions are met. See the Programming Fundamentals chapters for a complete description of using the IEN mode and Tables 3-4 or 4-3 for the arming requirements of the 69434A Event Sense card. The example assumes that W6 has been removed to allow individual arming of the card.
3. Reading the interrupting word. When the interrupt occurs the program will branch to a user specified routine to process the interrupt. This routine will read the input word without a gate. Reading with a gate will return the reference word and cause the interrupting input word to be lost. Thus, if the program needs to read the reference word, the input word should be read first, without a gate, and then the reference word can be read with a gate. Consult the 69434A Operating and Service manuals for details on modifying the manner in which the input word is stored.
4. Disarming the card. Once the card has generated an

interrupt, as indicated by IRQ being present in the return data word, the card should be disarmed. Generally the user will simultaneously disarm the card and load a new reference word, after which the card may be rearmed at any time. See the Programming Fundamentals chapters for a complete description of using the IEN mode and Tables 3-4 or 4-3 for the disarming requirements of the 69434A Event Sense card.

5-52 Pulse Counter Card, 69435A

5-53 The 69435A Pulse Counter card is a 12-bit up or down counter with a range of 0 to 4095. Carry and borrow outputs are available for use in cascading additional cards to extend the counting range or as alarms signals when using the 69435A as a pre-set counter.

5-54 Data Values. The data returned from the 69435A is in the range of 0 to 4095 (0000 to 7777 octal), and represents the number of pulses that have been counted (up pulses-down pulses). The data can also be interpreted as two's complement binary in the range of -2048 to 2047 by the use of the following equation:

Count=Data-(Data>2047)*4096

The data sent to the Pulse Counter card in the output mode to preset it also is in the range of 0 to 4095. If it is desired to treat the 69435A as two's complement, the data should be converted to the equivalent unsigned binary value prior to being sent using the following equation:

Data=Count+(Count<0)*4096

Note that when using the HP-IB the data must be converted to/from OCTAL before use in the above equations.

5-55 Programming the 69435A. A typical programming sequence for a Pulse Counter card located in slot 8, is illustrated in Example 12. The 69435A is first preset to a count of 100, next other operations are performed, these are left to the user to define), finally the Pulse Counter card is read and the data is interpreted as two's complement before being DISPlayed. TIME is not used when programming the 69435A as it does not have the necessary CTF circuitry.

5-56 Process Interrupt Card, 69436A

5-57 The 69436A Process Interrupt card provides TTL and open collector compatible edge detectors; one positive and one negative for each of 12 storage latches. Logic transitions lasting greater than 100 nanoseconds are detected, stored and can be used to generate an interrupt to the computer.

5-58 Data Values. The data returned by the 69436A has no special meaning attached to it by the Multiprogrammer (two's complement, sign magnitude, etc.), thus, it is up to the user to define the significance of the return data word. In many cases the data will be treated as individual bits, each with its own meaning. In any case the data as received by the computer will be in the range of 0 to 4095 (0000 to 7777 octal).

5-59 Programming the 69436A. The Process Interrupt card is generally only used in the IEN mode either singly or with other cards having IEN capability, and will generate an interrupt when any one of the 24 edge detectors is activated. There are five basic steps involved with programming the 69436A.

1. Initializing the edge detectors. This will reset all the edge detectors to a "0" state clearing any interrupts that may have occurred due to turn on conditions in the external circuits. Initialization is accomplished by sending a data word, in the output mode, with a "1" in all bit positions, 4095 decimal or 7777 octal.
2. Arming the card. This step will allow the card to generate an interrupt if any of the inputs change. See the Programming Fundamentals chapters for a complete description of using IEN mode and Tables 3-4 or 4-3 for the arming requirements of the 69436A Process Interrupt card. The example program assumes that W6 has been removed to allow individual arming of the card.
3. Reading the interrupting word. When an interrupt occurs the interrupt service routine will read the return data word without a gate, check for, and remove IRQ if present before taking whatever action is required by the system design in response to the interrupt. Testing for IRQ is important when using multiple cards in the IEN mode as it is the method used to determine which card caused the interrupt.
4. Resetting the card. After it has been determined that a particular card has generated an interrupt (IRQ present in the return data word), the card must be reset. This can be accomplished by sending the return data word (with the IRQ bit removed) back to the card as a data word. Only the bits that had interrupted prior to the read operation will be reset, therefore, any interrupts that occur after the read will be stored and will cause another interrupt the next time the card is armed. If further interrupts are desired, the card must be rearmed using the same procedure that was used for arming.
5. Disarming the card. If further interrupts are not desired, the 69436A should be disarmed. Sending a data word, in the output mode, to the card with "0" in all the bit positions, 0 decimal or 0000 octal, will disarm the card. After disarming, further interrupts will be inhibitted although the edge detectors are still active and will store any bit transitions that occur. Tables 3-4 or 4-3 contain the disarming requirements of the Process Interrupt card.

5-60 Example 13 shows a typical programming sequence for a Process Interrupt card located in slot 4 (it is assumed W6 has been removed). In this example after the interrupt the card is rearmed for further interrupts. If it is not desired to receive further interrupts, the substitutions indicated should be made.

5-61 Breadboard Input Card, 69480A

5-62 The 69480A Breadboard Input card provides a means of interfacing custom input circuits to the Multiprogrammer. It contains address and read circuits and a breadboard area for the

construction of the required circuits. Access to ISL, DST IEN, CTF, and IRQ is also provided to allow more sophisitcated input circuits to be built.

5-63 Data Values. The data returned from the 69480A will be in the range of 0 to 4095 (0000 to 7777 octal). The significance attached to the data is dependent on the user's circuitry, and can be interpretted in any way desired by proper programming.

5-64 Programming the 69480A. The program sequences that are required to program the Breadboard Input card are, in general, dependent on the custom circuitry and which control signals are utilized. Example 15 assumes that ISL, DST, and CTF are used in the breadboarded circuits. If the IEN and IRQ signals are used a programming technique for IEN mode will be used. See the Programming Fundamentals chapters for details on the various operating modes.

5-65 Resistance Output Cards, 69500A-69513A

5-66 The 69500 Series of Resistance Output cards provide a resistance output that can be used to program the output voltage or current of HP power supplies equipped with Option 040. Twelve magnetically shielded, mercury wetted, reed relays select the resistance values by modifying the value of a series string of high-accuracy, binary weighted resistors. There are three basic types of Resistance Output cards:

1. The 69500A is supplied without output resistors, allowing user to select and load their own resistors.
2. 69501A-69505A are single output, 12-bit resolution cards designed to program the voltage output of a single HP power supply equipped with Option 040.
3. 69510A -69513A are dual output, 6-bit resolution cards designed to program the current outputs of two HP power supplies equipped with Option 040.

5-67 Data Values (69501A-69506A). Table 5-1 lists the maximum resistance and resolution of each card. The data value necessary to program the desired output resistance can be easily calculated by dividing the required output resistance (in ohms) by the resolution of the card being used. The data value required to program an Option 040 power supply can be calculated by dividing the desired output voltage by .01 or .025 depending on the power supply model determined by Table 5-2.

5-68 Data Values (69510A-69513A). Table 5-1 lists the maximum resistance and resolution of each card. Channel 1 resistance output terminals are C and D while Channel 2 output terminals are A and B. The data value necessary the program a desired output resistance can calculated by the following steps.

1. Divide the output resistance for Channel 1 by the resolution of the card.
2. Divide the output resistance for Channel 2 by the resolution of the card and multiply the result by 64 (decimal). This will scale the value into the most significant 6 bits of the data word.
3. Add the values obtained in steps 1 and 2. The resulting value is the data portion of the data word that will

program both channels to the desired values. (HP-IB only: convert the data to octal.)

5-69 Calculating the data values used to program the current of Option 040 equipped power supplies is similar to the procedure used for resistance output. The difference is the desired output currents are divided by the current resolution determined by the power supply model and Table 5-2.

5-70 Programming the Resistance Output cards. The data calculated from the procedures given in the previous section is sent to the Resistance Output card to program it. The Multiprogrammer must be in the output mode (SYE on) and TME can be used to control the CTF flag allowing a delay between successive data words. Example 15 illustrates programming the output voltage of a 6271B power supply and Example 16 shows the procedure used to program the output current of two 6253A power supplies.

NOTE

The values programmed can only be multiples of the resolution of the card, whether it is being used for resistance, voltage, or current output. Thus, any value that is not an exact multiple of the resolution will be rounded to the nearest value that is.

5-71 Programmable Timer Card, 69600B

5-72 The 69600B Programmable Timer card provides a crystal controlled, programmable, one-shot output pulse. The duration of the pulse is determined by two factors: 1) the number of programmed time increments, from 1 to 4095; and 2) the selected period of the time increment, from 1 usec to 100 msec. Period selection can be accomplished by fixed jumpers on the card (or card edge connector) or be programmed with the use of the 69433A Relay Readback card.

5-73 Optional Jumpers. The output pulse is normally started when the data word is received, but it is possible by adding jumper W4 on the card to generate simultaneous pulses from a number of cards or to synchronize the output pulse with D/A converters using DTE. The 69600B may also be externally triggered if jumper W5 is added. The Programmable Timer card may also be used in the IEN mode if jumper W3 if removed.

5-74 Data Values. The data sent to the 69600B program the number of base periods that will be output. The time interval that will be output is the programmed value multiplied by the base period value. The range of the programmed value is 0 to 4095 (0000 to 7777 octal). The range of output times available is from 1 usec to 409.5 seconds.

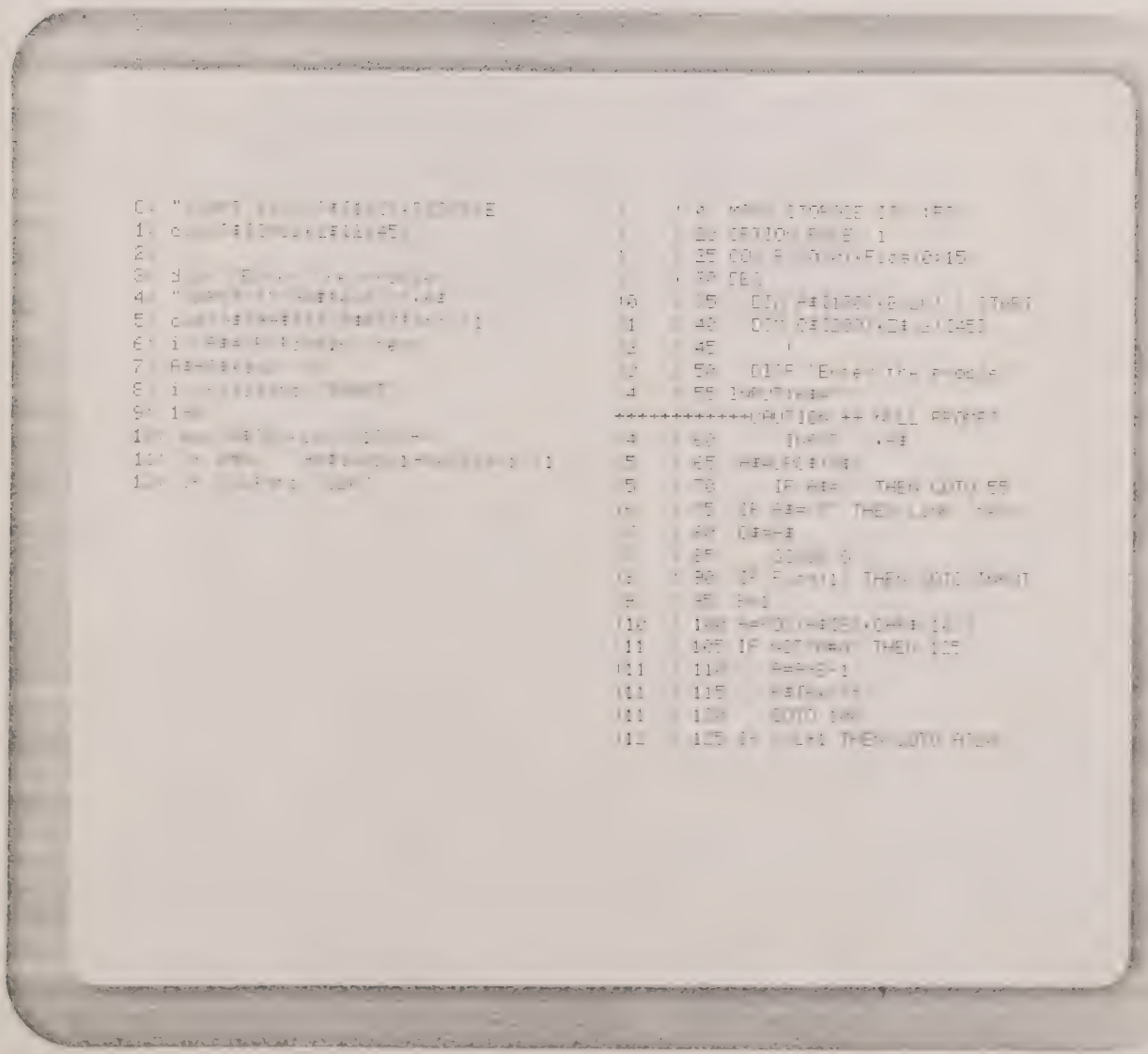
5-75 Programming the 69600B. The 69600B can be used in a number of operating modes as dicussed in the previous paragraph. Example programs for two of the operating modes, TME mode and IEN mode will be presented. When using the card in TME mode, the Multiprogrammer will wait until the output pulse is complete before other operations can be performed. The computer can wait

for the pulse to complete or an interrupt can be generated upon completion. Example 16 shows a Timer card in slot 9, being used with TME causing the computer to wait until the programmed pulse is complete.

5-76 Example 17 shows two Timer cards, located in slots 3 and 9, being used in the IEN mode. In this case the first card to "time out" will generate the interrupt to the computer. It can be imagined that the cards are externally triggered, and it is desired to know when the first card times out so another operation can be started. The return data word from the 69603D will contain the IR0 bit if it has timed out, allowing the computer to determine which card completed first so the appropriate action can be taken.

Hewlett-Packard Desktop Computer Software

HPL to BASIC Translator



HEWLETT hp PACKARD

Warranty Statement

Hewlett-Packard makes no express or implied warranty of any kind, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose, with regard to the program material contained herein. Hewlett-Packard shall not be liable for incidental or consequential damages in connection with or arising out of the furnishing, performance or use of this program material.

Use of the manual and tape cartridge (or flexible disk) supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only. Resale of the programs in their present form, or with alterations, is expressly prohibited.

Table of Contents

Chapter 1: General Product Description

Introduction 1

Chapter 2: Operating Instructions

General Operating Instructions 3
Translator Tape Organization and File Description 4
Detailed Operating Instructions — Program Set-Up 5
Changing the HPL Source 6
Changing the Input Buffer 6
Changing the HP-IB Select Code 7
Changing the Destination File Name 7
Changing the HPL Listing Selection 7
Changing the System Printer Select Code 7
Changing the Interactive Mode 8
Starting BASIC and HPL Line Numbers 8
Source and Destination File Checking 8
Additional Dimension Statements 8
Main Translator Operation 10
Cross Reference Tables and Program Termination 11

Chapter 3: Transferring the 9825 Program to Your Computer

Introduction 13
Program Transfer via 9825 Tape Cartridge 13
Program Transfer via HP-IB 14
Program Transfer via Flexible Disk 15

Chapter 4: Translating Linked Programs

General Considerations 17
How to Translate a Segmented Program 17
Other Linked Program Considerations 18
Branches to Locations Outside the Program being Translated 18
Subprograms and Functions 18

Chapter 5: 9825 Data File Translator

Introduction 19
Operating Procedures and Limitations 19
Program Files 19
tlist Option 19
Special Function Keys 19
Translation of Data Files 20
Exiting the Data File Translator 20
Limitations 20
Messages and Prompts 21
Prompts 21
Non-Error Messages 22
Error Messages 23

Chapter 6: Program Translator Limitations

Introduction 25
Limitations due to Source and Destination File Size 25
Maximum Size of HPL Program That Can Be Read from Tape 26
Maximum Number of HPL Lines Allowed 26
Maximum Number of HPL Label Characters Allowed 26
Limitation of the Number of Subprograms and Functions 26

Chapter 7: Optimizing and Running the Translated Program

Optimizing the Translated Program 27
Running the HPL Program 27
Correcting Translator Warnings and Cautions 28

Chapter 8: Translation Considerations — Mainframe Instructions

Labels 29
jmp Statement 30
Units Statement 30
r Variables 30
not Operator 30
Imbedded Assignments 31
HPL Flags 31
Arrays 32
min and max Functions 32

Cartridge Tape Commands 32
fdf and trk Statements 33
mrk and ert Statements 33
idf Statement 33
Other Tape Operations 33
rcf Statement 34
ldp Statement 34
ldf Statement 34

Chapter 9: Translation Considerations — Option ROMs

String ROM 37
The str Function 37
String Arrays 37
Matrix ROM 38
Plotter ROM 38
HP BASIC ROM Limitations 38
Extended Plotter Statements Using Output Statements 38
Axis Statements 39
psc Statement 39
csiz Statement 39
lbl Statement 39
Other Non-translatable Plotter Commands 40
Flexible Disk ROM 40
Mass Storage Unit Specifier 40
cat Statement 40
asgn and Assign statements 40
ens Operator 41
type Function 41
Other Non-translated Statements 41
General I/O ROM 41
Translation of fmt Statements to IMAGE Statements 41
Write Binary Statement wtb 42
Write Control Statement wtc 43
Read Status (rds) and Read Interface (rdi) Functions 43
List Statement 43

Extended I/O ROM 44
Octal/Decimal Operating Modes 44
Octal, Decimal, and Addition (add) Function 44
Interface Control Statements 44
Table Conversion and Parity Statements 45
Error Functions 45
Device Names 46
Buffered I/O 46
Timeout Statement 47
Interrupts 47
HP-IB Programming 48
Multiple Bus Addressing 48
Other HP-IB Statements 48
Advanced Programming ROM 48
For/Next Loops 48
Subprograms and Functions 48
Optimizing the Translation of Subprograms and Functions 52
Split and Integer Precision Storage Functions 53
Cross Reference Statement 53
Systems Programming ROM 53
Intelligent Terminal Instructions 53
End-of-Line Specification 53
Serial Interface Control Statements 53
HP BASIC Enhancements 54

Appendix A

Error Messages and Explanations 55
Translator Cautions 55
Translator Warnings 58
Translator Fatal Errors and Remedies 61

Subject Index 63

Chapter 1
General Product Description

Introduction

The HPL to BASIC translator is a high-level HP BASIC program. The program tape supplied will operate on the appropriate HP BASIC machine. The Translator allows direct 9825 program input and then executes a two-pass, line-by-line translation of that HPL source to create a BASIC program that will run on your HP BASIC computer.

The HPL to BASIC Translator is designed as an aid for the conversion of programs from the 9825 HPL language to the HP BASIC language. The two languages are different enough that the conversion process may not be complete, and the translation process will require manual correction and optimization after the translator has converted the majority of the code. The translator can translate up to 85-90 per cent of the HPL statements. You must hand-translate the remainder of the statements as well as optimize the translated code, where possible, to utilize the added power of the BASIC language. The translator itself should be viewed as the first step in the translation process, and, thus viewed, can be a very useful tool. The time saving of translating a program versus re-writing can be up to 1:10.

Two listings are produced by the translator. The HPL source is listed during the first pass, and the translated BASIC is listed during the second pass. Any statements that are not translated, or statements where the code might need checking, are flagged on the BASIC listing. The BASIC code is also listed into the file selected by the operator. The translated BASIC program can be entered into the computer memory by executing a **GET** on that file.

NOTE

The translator is a tool to assist in the conversion of HPL programs to HP BASIC. It normally will not translate the programs 100 per cent. Therefore, the translation process requires the following three steps:

- Running the translator to convert the majority of the code and create the output file.
- Editing the translated program to correct the syntax errors noted by the translator and the BASIC syntaxer.
- Running the program to verify proper operation and detect any run-time errors not detectable from the syntax.

The translation process is not complete until all these steps have been taken.

A detailed breakdown on the capability of the translator is given later for all the mainframe and option ROM instructions.

Chapter 2
Operating Instructions

General Operating Intructions

To run the Translator, the following are needed:

- HP Desktop Computer with 64K minimum Read/Write memory.
- A hardcopy output device such as the 9866, 9871, or 2631 printer.
- The HP 9885M Flexible Disk can be used in conjuction with the Mass Storage ROM to increase the operating speed.

To run the Translator program, execute the following statement:

```
LOAD "INTRO",10
```

The **INTRO** routine will give you instructions on how to set up the mode of 9825 source transfer, how to set up the Destination file name, and how to set up the other program parameters. Each of the options is explained later in this section.

The **INTRO** routine will link you automatically to the program file translator or to the data file translator.

A utility program named **MAKETP** is supplied for use in copying the translator files to a flexible disk. The translator program will run significantly faster using the flexible disk, and the source and output files can be much larger. To run the **MAKETP** program, execute the following statement:

```
LOAD "MAKETP",10
```

The routine will ask for the source and destination Mass Storage Unit Specifiers, and will then copy the files from the translator tape to the disk.

Translator Tape Organization and File Description

The HPL to BASIC Translator is a series of programs that translate an HPL source to a BASIC program. The various segments of the translator are:

- **INTRO**: Introduction and system set-up. This is the program that is loaded and run to start the translator. All other program segments are automatically called by **INTRO** (except for the routine **MAKETP**).
- **PASS1**: First pass of the translation process. The majority of the functions and statements are translated during this pass.
- **PASS2**: Second pass of the translation process. The conversion to BASIC is completed and the BASIC program is listed. Any error messages are also listed.
- **PASS1B**: Additional pass between PASS1 and PASS2 used for the complete translation of subprograms and functions.
- **LISTER**: Routine to print out several different cross-reference tables to aid in program debug and optimization.
- **TAPE**: Routine to read the 9825 cartridge tape using the internal tape drive. The 9825 program file is also decompiled and listed.
- **READIO**: Subprogram driver that is appended to **PASS1** if the program input is via HP-IB. The I/O ROM is needed to input via HP-IB.
- **TCMPL**: Routine to translate 9825 data files on a 9825 tape to HP BASIC data files. This routine also lists key files and can provide a **tlist** of the 9825 tape.

The only operator interaction is in setting up the program parameters, selecting the track and file number of the 9825 program tape to be translated, and in selecting, after the translation process is complete, which cross-reference tables are desired. An Interactive Mode is allowed during the processing of subprograms and functions during **PASS1B**, if desired, to increase the overall effectiveness of the translation of those segments.

Several other files exist on the translator tape. Their purpose is as follows:

- **INBUF**: This is used as the default file to store the HPL source as it is read in from the 9825 tape. It is also used during **PASS1B** to buffer the subprogram program segments.
- **OUTBUF**: This is used as the default file to store the translated BASIC program. At the completion of the translation process, the translated program can be loaded into memory by performing a **GET** on this file.

NOTE

The files selected for use as the source and destination can be changed during the introduction phase.

- **ADD-FN**: Contains subprograms and functions that simulate several HPL functions that do not have a direct translation (such as **stf**, **fti**, **add**, etc). These subprograms and functions are added automatically at the end of the translated program if needed.
- **INTROD**: Data file used by several portions of the translator containing the set-up parameters, the HPL to BASIC mnemonic translation table, and the error messages.
- **MAKETP**: Program to transfer the translator files to a flexible disk.

Detailed Operating Instructions — Program Set-Up

The operation of the HPL to BASIC Translator program is as follows:

A. Load the **INTRO** program and run it by executing:

LOAD "INTRO",10

B. The system will display the setting of the following parameters:

- HPL Source (File Name, **9825**, **HP-IB**, or **KEY**)

  Buffer File Name if the input specified is **9825**.

  Select Code if the input specified is **HP-IB**.
- Destination File Name
- HPL/Data File Translator Listing Selection
- System Printer Select Code
- Interactive Mode Selection
- BASIC/HPL Starting Line numbers

If all the settings are satisfactory, then press the special function key 0 (**K0**) for program translation, or Key 1 (**K1**) for data file translation. If any of the parameters require changing then select the appropriate special function key. The program will then explain what to do to

modify that parameter. After all the parameters are set properly, press the special function key **K0** or **K1**.

The data file translator can be loaded at this time. Note that the settings of the system printer and the listing selection flag are used in the data file translator.

The explanation of each parameter follows:

Changing The HPL Source

Four options exist for HPL source input:

- Direct input from a 9825 program tape: The translator, using the program **TAPE** and a special binary, will read automatically the 9825 program file from the 9825 tape, decompile it, and list that source on the system printer and also into the designated HP BASIC data file. This is the most convenient method for transfer. To select this, type in **9825** and hit **CONTINUE**.
- Input from a mass storage data file: The 9825 program must exist in a serial access file in string form, stored as 1 program line per string. The section on the flexible disk transfer explains how to use the 9825 and the disk to create such a file. To select a file as the input, type in the file name. Note that the names **9825**, **KEY**, and **HP-IB** are reserved for the other methods of transfer and thus cannot be used as file names.
- Transfer via HP-IB: The program can be transfered via HP-IB from a 9825 list. HP-IB transfer is covered more in the next section. To enter via HP-IB, type in **HP-IB** and hit **CONTINUE**.
- Entry via keyboard: The program or program segments can be keyed in from the keyboard. Note that no syntax checking is done, so the syntax of the entered line must be exactly the same as the 9825 listing. To enter via keyboard, type in **KEY** and hit **CONTINUE**.

Changing the Input Buffer

If the input specified is **9825** the system will request the name of an input buffer file. Reply with the name of a valid mass storage data file. If the file is not on the default mass storage device, then append to the name the **msus** (**m**ass **s**torage **u**nit **s**pecifier). Examples of valid replies are:

```
INBUF
INBUF:F8,1
```

Changing the HP-IB Select Code

If the input source specified is **HP-IB**, then the system will ask for the select code of the HP-IB interface card.

Changing the Destination File Name

To change the destination file name, reply with a valid file name. If the file does not exist on the default mass storage device, then append the **msus** to the file name. Examples of valid responses are:

```
OUTBUF
OUTBUF:F8,1
```

NOTE

The Input Buffer and Output Buffer must be accessible at the same time.

NOTE

The translator tape is shipped out with two files, **INBUF** and **OUTBUF**. These files are sufficiently large to allow the translation of at least an 8K HPL program. The default buffer and destination file selections are set to these file names. Refer to the section on translator limitations for information on translating long programs and file size requirements.

Changing the HPL Listing Selection

The HPL source can be listed on the system printer during the process of the translation by replying **Y** or hitting **CONTINUE**. To inhibit the HPL listing, reply **N**. This selection is also used to suppress the listing during the data file translation.

Changing the System Printer Select Code

The select code of the printer for the HPL and BASIC listings can be input. To list the program to the CRT, use select code 16. An HP-IB device can be used for the system printer. Examples of the proper format are shown on the CRT. This selection also determines the listing device for the data file translator.

Changing the Interactive Mode

The interactive mode can either be enabled or disabled by replying **Y** or **N**. The interactive mode is invoked in PASS1B when the boundaries of subprograms or functions are not well defined. If the interactive mode is not selected, the program will make an assumption and continue. If the interactive mode is selected, the program will prompt you and ask you to make a decision.

The interactive mode also allows you to specify if an un-dimensioned string variable is an array or simple variable.

The interactive mode is covered more in the section on Advanced Programming.

Starting BASIC and HPL Line Numbers

The translator allows you to specify the starting line number for the translated BASIC program. With the default setting, the BASIC program starts with line number 20. This feature is useful when translating linked programs as the linked programs can be translated starting with different line numbers and then be loaded into the computer memory concurrently.

The translator also allows the starting HPL line number to be specified when reading the 9825 program tape. This is used if a portion of a linked program is stored in a file, and the portion does not start with HPL line number 0.

Source and Destination File Checking

After the translation parameters are set up, the system will check the destination and source files. If they are not available, the system will print a warning and allow you to update the file name. The source and destination files must be available during all phases of the translation process. However, the source and destination files can exist on a separate cartridge and be switched in and out of the tape drive. In the following program segments the operator will receive a prompt if the proper file is not available to the program.

Adding Additional Dimension Statements

The system then asks you if this is a linked program. If not, then hit special function key **K0** to continue. Otherwise, hit key **K1**. This feature allows additional dimension statements to be input.

Normally this feature will NOT be used. However, if the following circumstances exist, additional dimension statements should be used:

- There are either string variables or arrays used in the program that are not dimensioned in the program itself (i.e. they are dimensioned in a program that then links to this program). All string and array variables must exist in a dimension statement to be properly converted and placed into a COMmon statement if needed.

- If this program is to be called by another program through the **LOAD** statement, all variables which must retain their value must be placed in a COMmon statement in both routines. The COMmon statement should be created after the first segment is translated and then entered into this program.

In entering additional **dim** statements, they can either be entered by hand or the file name can be entered. Individual statements are entered by typing in **dim** or **COM** followed by the dimension statement (follow the format shown on the CRT). The name of a file containing **DIM** or **COM** statements from a previous segment can be entered instead, and the translator will automatically search out all **DIM** or **COM** statements. The file name entered cannot contain **DIM**, **COM**, **dim**, or % since they are used as keywords.

HPL comments can be added at the start of the program by using the HPL free-syntax operator % before the line.

Main Translator Operation

After the introduction and set-up stage, either **PASS1** or the **TAPE** routine is loaded depending on the source of the HPL code:

If the input was specified as **9825**, then the **TAPE** routine is loaded. The **TAPE** routine will prompt you for the track number and file number of the 9825 program file. It will then prompt you to insert the 9825 tape and hit **CONTINUE**. After the tape is read, you will be prompted to reinsert the translator cartridge tape. If flexible disk is being used, simply hit **CONTINUE**. The HPL program will then be decomplied, listed if desired, and saved in the buffer file. **PASS1** will then be called directly.

For input from **HP-IB**, **KEY**, or a File, the **PASS1** routine is loaded. If the input is **HP-IB**, then the **READIO** subprogram is also loaded.

After the introduction phase (**INTRO**) and the 9825 tape has been read, if desired, **PASS1** is loaded into the computer and the line-by-line translation begins. The intermediate translated line is displayed in the display area of the CRT.

After the HPL source has been completely converted by the first pass, either **PASS1B** or **PASS2** is loaded depending on whether any subprograms or functions exist in the HPL program. If **PASS1B** is loaded, all the subprograms and functions are relocated to the end of the program, subprogram and function headers are created, and all variables that are used in subprograms are put into COMmon. The program then calls **PASS2**.

NOTE

PASS1B requires either a file named **SRC1** or the file **INBUF** to buffer the subprogram modules during the rearrangement process. If the file **SRC1** exists, it is used. If not, the file **INBUF** is used. The translator tape, in order to make **INBUF** as large as possible, does not contain a file **SRC1**. If the flexible disk is used or the input and output files are on another tape, then a file **SRC1** can be created and used. This allows the input buffer file to remain unchanged in case the HPL source must be used again.

PASS2 completes the translation process, printing each line including any warnings or errors, and listing the BASIC into the destination file. Any translator warnings appear above the line containing the error. If the line is subdivided due to the BASIC translation or an implied assignment, the warning may be printed above the first statement of that sequence.

Cross-Reference Tables and Program Termination

At the end of **PASS2**, the translation in complete. The routine **LISTER** is automatically called. This routine allows the following cross-reference tables to be printed:

- HPL Line Number to BASIC Line Number
- HPL Labels to BASIC Labels
- Format to Image Statements
- Format Statement References
- Subprogram and Function Calls

In addition, the translator can be restarted, or the translated program can be fetched. After the fetch is complete, due to the linkage in BASIC, the translated program may be run and an error may occur (this is normal).

When the program is fetched, all statements that do not syntax correctly are printed on the system printer and converted into comments. Remember that the translation process is not complete until the translated program has been fetched and all the errors corrected. During the correction phase, the BASIC listing produced by the translator must be consulted as many warnings produced by the translator are not flagged when the **GET** is executed.

NOTE

When the program is being fetched, all option ROMs needed to run that program (such as I/O, Plotter) must be in place. Otherwise the translated lines will not syntax correctly and will be converted into comments.

Chapter 3
Transferring the 9825 Program To Your Computer

Introduction

Several alternatives exist for the transfer of the 9825 program to your computer. In each case, the program must be transfered as ASCII string characters. Each of the following methods is aimed at that result. Choose the easiest method for your circumstance.

Program Transfer Via 9825 Tape Cartridge

Direct reading of the 9825 program tape is the most direct method of program transfer. This method requires the special binary program that is present on the translator tape in the **TAPE** program.

The **TAPE** routine is called by the **INTRO** routine whenever **9825** is specified as the source of the HPL program. You are prompted for the track number and file number of the HPL program file. You are then prompted to insert the 9825 tape into the drive, and the tape file is then read.

After the program file is read, the code is de-compiled into the same format as a listed HPL program. This is necessary because the 9825 program file is stored in an internal compiled format.

As the program is decompiled, it is listed, if desired, and stored into the buffer file named in the **INTRO** routine. When the entire HPL program has been decompiled, the **PASS1** routine is called.

Program Transfer Via HP-IB

The program can be transfered from the 9825 to the computer via HP-IB. The two machines are connected by 2 HP-IB 98034[1] interface cables. While the program is being translated during **PASS1**, the 9825 must list the program to the interface card.

To transfer via HP-IB, do the following:

- The interface card that goes into the computer must be modified so that it is HP-IB bus device #20 and is NOT the system controller. Refer to the 98034 Interface Manual for instructions.
- Specify during the **INTRO** routine to transfer via **HP-IB** and specify the proper select code.
- After **INTRO** is loaded, load the program into the 9825 and execute the following statement:

  list #720

NOTE

In order to do the list on the 9825, all Option ROMs must be installed and the flexible disk, if accessed by any of the instructions, must be active. Otherwise, list errors will occur on the 9825.

NOTE

If the system printer is also an HP-IB device, two HP-IB cards must be used within your computer and the busses must be kept separate.

---

1 The HP-IB interface card that is plugged into your computer MUST be an HP 98034A **Revised** interface card

Program Transfer Via Flexible Disk

The program can be transfered by using a flexible disk data file. A disk that is initialized on the 9825 can be directly used on your computer (Note that a disk that has been initialized on your computer lacks the necessary prompts to work on the 9825).

The data file must be opened on the 9825 and then the HPL program must be stored in it as a series of strings. To do that, append the following program to the HPL program to be translated while in memory and then execute a **run "STORE PROGRAM"** :

```
0: "STORE PROGRAM":dim A$[100],F$[20]
1: buf "LINE",100,1;ent "Enter the Dest. File Name",F$
2: ent "Enter the Last Line Number to be stored",E
3: asgn F$,1
4: for I=0 to E
5:  list#'A', I,I
6: red "LINE",A$;dsp A$
7: sprt 1,A$
8: next I
9: dsp "HPL Program Saved";end
10:  "A":ret"LINE"
11: end
```

The 9825 must be configured with all necessary option ROMs in order to create the list. The 12 line program can be saved in a special file and loaded by executing a **get**. This simplifies the creation of the HPL source file.

Chapter 4
Translating Linked Programs

General Considerations

The translation of linked HPL programs is somewhat more difficult than the translation of a single stand-alone program. A program that was segmented in HPL may not, due to the increased memory size of your computer, have to be segmented. The segmentation might change. In addition, variables that are dimensioned in the first segment will need to be dimensioned for the translator when translating following segments so that the string variables can be converted correctly and COMmon statements correctly created. Thus, the translation of a linked or segmented program is not as straight forward as for the stand-alone program, but the translator contains some tools to make the task easier.

How To Translate A Segmented Program

In general, the first segment that should be translated should be the segment that dimensions the variables. After that segment is translated, it should be edited to correct the dimension statements. If the following segments contain subprograms and some variables are used much more often than others, the COMmon statement should be generated at this time to optimize the placement of variables within the COMmon statement.

When the following segments are translated, allow the translator to insert the dimension statements from the first file into the program being translated. To do that, hit special function key **K1** when asked if it is a linked program, then enter the name of the file that contains the first segment. The translator will search that file and insert all the dimension statements at the front of the segment being translated.

After the translation of each following segments, the dimension statements can be deleted. If the COMmon statements were updated, they should be updated for the first segment so that all programs use the same COMmon statement.

When all segments are completed, check and make sure that all the COMmon statements are the same.

There are two ways to link BASIC programs, with the **LINK** and with the **LOAD**. The **LINK** offers the same capability as the 9825 (although slower) with the ability to locate the source being entered anywhere. Thus, a segment can be retained in the present program while overwriting a portion with higher line numbers. The **LOAD** executes faster, but the program overwrites all of the former program. Variables that must be preserved between segments (including single variables) must be stored in COMmon (another reason for creating the COMmon statement).

Other Linked Program Considerations

Branches To Locations Outside The Program Being Translated

If a branch occurs to an HPL line that is not in the segment being translated, the translator cannot determine what to convert it to. It replaces the line as follows:

```
gto 560
```

with

```
GOTO H-560
```

After the translation, those lines will need to be corrected, but that is easily done by doing line-number cross-reference tables for each segment as it is translated.

To aid in this problem, it may be desirable to translate several program segments at the same time.

Subprograms and Functions

In order to translate a section of code as a subprogram or a function, it is necessary for that section to be called within the program being translated. If it is not called, it will not be translated as such. It might be necessary to add additional statements on the end of the program to call the subprograms and functions so they will be correctly translated.

BASIC requires all subprograms and functions to be placed at the end of the program. This creates a problem if a series of subprograms or functions are used by several linked programs and thus kept in the lower part of a program. If possible, they should be converted into simple subroutines. Otherwise, they will have to be saved as a data file and loaded at the end of the program each time a link is performed.

Chapter 5
9825 Data File Translator

Introduction

This program provides for the translation of 9825 files to HP BASIC Mass Storage files. The 9825 file is read, and its contents are translated and/or displayed. The file type is used to determine the action taken. File types 0,1,4 and 6 (Binary Program, Read/Write memory, Program, Null or Empty) are indicated with a message. Type 5, Special Function Key, files are printed on the print device in a format that shows the key definition and key number. File types 2 and 3 (numeric data, data file including string) are translated according to a **dim** and **rcf** or **ldf** statements provided by the operator.

Operating Procedures and Limitations

Program Files

The main HPL to BASIC Program Translator should be used to translate HPL programs.

tlist Option

To obtain a **tlist** of the files on a 9825 tape, LOAD and RUN program INTRO. Select the data file translator by pressing special function key 1. When the translator is loaded use special function key 1 to select this option.

Special Function Keys

To obtain a listing of the special function key definitions (type 5) - LOAD and RUN the program **INTRO**. Select the data file translator using special function key 1. After the data file translator is loaded use special function key 0 to select data file translation. Answer the questions about track number (0 or 1) and the file number. Install the 9825 tape, and press **CONTINUE**. The listing will be printed on the print device (PRINTER IS n) as specified by **INTRO**.

HPL keys definitions containing an immediate execute for one statement only, or an immediate continue can be directly translated to your computer's key. If the key definition involves several executable statements, it is replaced by a BASIC level subroutine and an "ON KEY" statement within the program linking to that subprogram.

Translation of Data Files

To translate data files (types 2 and 3) - LOAD and RUN program **INTRO**. Select the data file translator by using special function key 1. After the data file translator is loaded use special function key 0 to select data file translation. Answer the questions about the track number (0 or 1) and the file number (0<=n). Install the 9825 tape and press **CONTINUE**. When the tape file has been read a prompt to switch back to the computer tape will appear. This prompt includes a query for the computer file name where you want the translated data to be placed. If the file does not exist, the program will create the file automatically. The program will then accept and syntax check a **dim** statement. The **dim** statement must not use variables to specify the size of arrays (not acceptable: dim A[B]). When the **dim** statement is accepted the prompt for the **rcf** or **ldf** statement will appear. In response to this enter the statement that would normally be used to reference the data in the file. The rcf/ldf statement syntax and data type will be checked as the data is translated. Translated data is displayed on the CRT as the translation takes place. It is also printed on the system printer if enabled during the **INTRO** routine.

Exiting the Data File Translator

To exit the data file translator and return to **INTRO** use special function key 2.

Limitations

1. The dimension used in the dim statement must not use a variable (**dim A[B]**)
2. HPL arrays of more than 6 dimensions can not be translated as a single array. The HP BASIC limit is 6 dimensions. You must segment these arrays in order to translate them.

Messages and Prompts

Prompts

Prompt	Response
Enter track number.	Type the track number 0 or 1, then terminate input with **CONTINUE**
Enter file number.	Type the file number (0<=n) then terminate input with **CONTINUE**
Install the 9825 tape and hit **CONT**	Exchange the tapes when necessary and press **CONTINUE**.
Install computer tape. Enter the destination file name.	Exchange the tapes when necessary. Type in the destination file name and terminate input with **CONTINUE**.
Enter the HPL dim statement used in the program that accesses this file.	Type in the HPL **dim** statement and terminate input with **CONTINUE**. If r variables are to be translated then only press **CONTINUE**.
Enter the HPL rcf or ldf statement used in the program that accesses this file.	Type in the HPL **rcf** or **ldf** statement and terminate input with the **CONTINUE** key. Include in the statement the file number (rcf 0,A$). This is required for syntax only.
Does this file contain r variables?	This prompt is given when the **dim** is not given. If r variables are to be translated then type Y and **CONTINUE**. If not type N and **CONTINUE**.
Select Special Function Key for action desired.	A display of available program action is printed on the CRT and corresponding Special Function Key numbers. You should make a selection.

Non-Error Messages

(title message): 9 8 2 5 T A P E F I L E T R A N S L A T O R

** File translation	This message is printed upon selecting file translation.
** tlist	This message is printed upon selecting 9825 **tlist**.
TRACK number	Is printed after the track number is accepted
FILE number	Is printed after the file number is accepted.
Length of file is X bytes.	This message is printed and gives the length in bytes of the 9825 file.
The selected file is a program file.	This message is printed and explains the file type.
The selected file is a Special Function key file.	This message is printed and explains the file type.
The selected file is a string and numeric data file.	This message is printed and explains the file type.
The selected file is a numeric data file.	This message is printed and explains the file type.
The selected file is a Binary Program file.	This message is printed and explains the file type.
The selected file is a Read/Write Memory file.	This message is printed and explains the file type.
The selected file is an empty file.	This message is printed and explains the file type.
The selected file is a null file.	This message is printed when the file header indicates an absolute file size of zero. NOTE: The last file on each track would be of this type.

File translation - Reading 9825 tape.	This message is displayed while reading the 9825 tape during file translation.
tlist - Reading 9825 tape	This message is displayed while reading a 9825 tape during a tlist.
** BASIC destination file is _____	This message is printed upon entering the HP BASIC destination file name.
Loading INTRO.	This message is displayed upon termination of the data file translator.

Error Messages

Improper track number -- Enter track number.	This error message is displayed when the entered track number was not 0 or 1. Enter a 0 or 1 and **CONTINUE**.
Selected file not found--Is 9825 tape installed? Correct and hit **CONT**.	This error message is displayed when the requested file number probably does not exist or the tape may be defective.
End of tape hole detected. To re-try, hit **CONT**.	This error message is displayed when a hole is detected in the 9825 tape. These holes are at both ends of the tape.
Checksum error--Is 9825 tape installed? Correct and hit **CONT**.	This error message is displayed when a checksum error is detected in the file header or in the file body.
The selected file would exceed the available memory space.	This error message is displayed when reading a file of more than 25,000 bytes.
The absolute file size is 0. **CONT**.	This error message is displayed when the file header indicates an absolute file size of zero. NOTE: The last file on each track would be of this type.

Is 9825 tape installed? Correct and hit **CONT**.	This secondary error message is displayed when a possible cause of error was the presence of a BASIC tape.
The BASIC tape is write protected. Correct and **CONT**.	The protect switch on the cartridge tape. Remove and unprotect. Reinstall and **CONTINUE**.
File ———————— is protected. Correct or enter new name and hit **CONT**.	The destination file name was not found or the file is protected. You should CREATE the file or select another file name. Terminate input with **CONTINUE**.
Improper dim statement. Hit **RECALL**, correct and **CONT**.	This error message is displayed when a syntax error is detected in the dim statement. **RECALL** the last line entered, correct and **CONTINUE**.
Improper rcf or ldf statement. Hit **RECALL**, correct and **CONT**.	This error message is displayed when a syntax error is detected in the rcf or ldf statement. **RECALL** the last line entered, correct and **CONTINUE**.

Chapter 6
Program Translator Limitations

Introduction

There are several limitations to the size of HPL program that can be translated. In general they are:

Limitations Due To Source and Destination File Size

The size of the input, output, and subprogram buffer file (if it exists) determines how large an HPL program can be translated. In general, the input file should have 8 records for every 1K of HPL program length (i.e. a 2K program would need **INBUF** to be 16 records long). The output file should contain 16 records for every 1K of HPL program. The increase in size is due to the larger program size needed for BASIC (an average BASIC program will take 2 to 3 times as many bytes as the corresponding HPL program) and for internal representation during the translation process.

If the file **SRC1** exists for buffering subprograms, it should contain about 8-10 records for every 1K of HPL program length. The above lengths are maximum numbers for the vast majority of cases. Most programs will translate with much smaller files. If the input is via 9825 tape, the translator will check the length of the files to make sure that they are large enough.

The translator tape, when shipped, contains large enough files to translate an 8K program. If a larger program is to be translated, then a flexible disk can be used or the **INBUF**, **OUTBUF** and **SRC1** files can be created on a separate tape and switched in and out. The suggested size for these files would be (for a 16K HPL program):

File	Suggested Size
INBUF	128 records of 256 bytes/record.
OUTBUF	256 records of 256 bytes/record
SRC1	160 records of 256 bytes/record

Maximum Size of HPL Program That Can Be Read From Tape

The maximum size of HPL program that can be read from a 9825 tape is 20K bytes. This is a limitation due to having 64K bytes R/W memory. Longer programs must be segmented in order to translate.

Maximum Number of HPL Lines Allowed

The maximum number of HPL lines allowed is 750.

Maximum Number of HPL Label Characters Allowed

The maximum number of HPL labels referenced by **gto**,**gsb**, and subprogram and function calls varies depending on the number of characters in each label, but the total number of characters allowed in the symbol table **Label$** is 1400(each HPL label requires 4 bytes of cross-reference information plus the number of characters in the label). An HPL label is entered into symbol table only if it is referenced in a **gto**, **gsb**, **cll**, function call, or a **dev**, **equ**, or **buf** statement. If a program overflows the symbol table (as explained in Appendix A), then start the translator again and sometime during the **INTRO** routine change the value of the variable **Label—limit** from 16 to a lower value. Do that by executing the following statement after **INTRO** has started:

```
Label—limit=X
```

where X is a number less than 16 (see Appendix A for details).

This will truncate the HPL labels more but will allow the program to be translated.

Limitation of the Number of Subprograms and Functions

The maximum number of HPL subprograms (not to be confused with mainframe subroutines) and functions allowed is 100.

Chapter 7
Optimizing and Running the Translated Program

Optimizing the Translated Program

There are some cases where BASIC can offer some improvements in the length of code over HPL. Some of the things to look for are:

- Places where a **jmp** followed by a series of **gto**s or **gsb**s could be replaced by the computed **GOTO** or **GOSUB**.
- Places where the built-in power of the matrix commands could be used.
- Places where, instead of accessing a number of HPL files, a single file could be used with random access.
- Places where the added power of the I/O ROM could be used.

Running the HPL Program

When running the HPL program, there are various defaults that the translator does not put into the program because they may be carried from one program to another. Among them are:

- The 9825 powers up with **fxd 2**. HP BASIC powers up with **FIXED 0**.
- The 9825 powers up in **deg** mode. The translator inserts a **DEG** statement in the front of each program, but each subprogram reverts back to **RAD** mode.
- The printer for the 9825 **prt** and **spc** is always the strip printer. HP BASIC powers up with the printer select as 16, but that can be modified with the **PRINTER IS** statement.
- The free-form printing and display format of the 9825 **prt** may be somewhat different than that for BASIC.

Correcting Translator Warnings and Cautions

The translator attempts to issue warnings with every statement that is not translated or that contains a potential bug. For example, every **not** is flagged, although only a fraction ever require any changing. It is your resonsibility to verify the proper operation of the program after translation. HPL contains some subtle differences that can never be detected during the translation process, and thus the program should be verified for proper operation.

The BASIC Utility Library supplied with your computer contains some BASIC level routines that operate on a program when stored as a data file. Among those routines that will be found useful are:

- SEARCH: Routine that searches for the occurrence of a ASCII string within a program. This is most useful for finding occurrences of labels, line numbers, strings, or arrays. It is not especially useful in finding occurrences of single variables.
- SEARPL: Routine that does the search as described above and in addition allows that string to be replaced by another string. A second data file is required. This routine will be useful in renaming strings, arrays, or **r** variables to make the program more readable. It is not very useful in renaming simple variables (such as in renaming the variable **A** to **Printer—sel**).
- XREFAL: Routine that does a complete cross-reference on the program, including simple variables, arrays, strings, and line labels and references.

Chapter 8
Translation Considerations — Mainframe Instructions

Labels

Labels in HPL are characters within quotes. Any character is allowed. Labels in BASIC consist of a capital letter followed by 0 to 14 lowercase letters, numbers, or the underscore. Labels are converted according to the following rules:

- A label not referenced by a **gto**, **gsb**, **cll**, or a function call is converted to a BASIC remark (preceded by a **!**)
- The HPL label is truncated to 16 or fewer characters for cataloging and comparison (depending on the value of the BASIC variable **Label—limit**). If the first character is not a letter, the letter **A** is inserted in the first position. All following characters are numbers, lower case letters, or the underscore character.
- The formed BASIC label is compared against all existing BASIC labels stored in the symbol table. If the label already exists with a different HPL label, numbers are added on the end to make it unique. In the unlikely event that a label cannot be made unique, an error message is printed.
- HPL allows the legal use of multiple-defined labels which is not allowed in BASIC. Only the first encountered line label is left as a line label.

The **LISTER** routine enables you to print out the label symbol table including the HPL label, the corresponding BASIC label, the line number where it was found, and the type of label.

jmp Statement

The **jmp** statement cannot be fully translated, since the **jmp** may depend on program activity. A **jmp** followed by a simple number is translated similar to a **gto** statement. Otherwise, it is not translated and will have to be manually translated to a BASIC computed **GOTO** or **GOSUB**.

In some extreme cases where a general **jmp** occurs (such as in **jmp G-500**) the jmp expression will need to be analyzed to determine the bounds of the expression. This may have to be determined by running the translated program.

Calculated gosub branching with the **jmp** is also not translated. Whenever a **jmp** instruction is noted, it should be checked for a computed **GOSUB** as it is not specifically flagged. The **GOSUB** statement appears after the **jmp** statement in the BASIC listing.

units Statement

The **units** operator is not translated. If the **units** function is required, the translated program will have to be modified to keep track of the current units in a string and then display that string.

r Variables

HPL **r** variables are converted to an array **Rv**. During the first pass the source is searched for the maximum value of r variable used. This value is then used to set up the dimension statement. If the program contains an instruction such as:

```
rA→B
```

where the limit of r variables cannot be determined, a caution is printed at each location in the program where the maximum value of r variables was inserted.

not Operator

The mathematical priority of the **not** operator is different in HPL and BASIC. Although this is not corrected, each occurrence of the **not** operator is flagged.

Imbedded Assignments

Imbedded assignments which can occur in HPL are translated into as many lines as necessary in BASIC. In the cases where the variable is used in the same statement as where it is assigned, the results might not be correct due to the way the 9825 interprets the line. For example:

```
(25→A)+1→B translates to A=25
                         B=(A)+1
```

giving a result of 26 in both HPL and BASIC

```
A+(25→A)→B translates to A=25
                         B=A+(A)
```

giving a result of 50 in both HPL and BASIC, but

```
1A+(25→A)→B translates to A=25
                           B=1*A+(A)
```

giving a result of 25 in HPL and 50 in BASIC.

The problems resulting from defining a variable and then using it later in that statement are not resolved within the translator, and you should be prepared to resolve any conflicts. No error is produced.

In many cases, because BASIC allows only one statement per line, one HPL line is translated into several BASIC lines. This may affect the operation because more end-of-line cycles are produced. An end-of-line interrupt can now occur in BASIC where in HPL it could not.

HPL Flags

Flags are used on the 9825 which do not exist in BASIC. The Translator converts these to an array **Flag**. Setting and clearing Flag14 on the 9825 corresponds to setting and disabling the defaults. The default values correspond very closely. In the 9825, however, Flag13 and Flag15 have special meaning that cannot be translated. Therefore, if an instruction references either of those flags, a warning is produced.

If an instruction accesses two flags on a **cmf** statement, two BASIC statements are produced. If the same flag is referenced twice, it is complemented twice instead of once as in HPL.

Arrays

BASIC arrays are limited to 6 dimensions. HPL arrays are not limited. The execution of the **GET** will detect that error.

HPL allows variable dimensions within a dimension statement, such as:

```
dim A[N]
```

Such statements are not allowed in BASIC except in subprograms. These statements are flagged, and you will need to dimension them to the maximum expected value.

All string variables and arrays that are used must appear in a dimension statement. This is necessary to properly convert string arrays and to properly create the COMmon statements.

min and max Functions

HPL allows the inclusion of entire arrays in the **min** and **max** statements. BASIC does not allow that and the line is flagged upon the execution of the **GET** on the destination file.

Cartridge Tape Commands

The Cartridge Tape System of the 9825 is file-by-number and is more primitive than the file-by-name approach of BASIC. The Translator changes all tape commands to work on the file-by-name approach even though some statements do not translate and all will require manual changing after the translation is complete.

When a file name is needed in a translated statement, a warning is issued showing the file number that needs replacement. The program inserts the characters:

"FILE--" followed by the file number accessed

You must insert the new file name after translation. Note that the file name can be a string variable so that different files can still be accessed from one instruction. As an example, both of the programs below reference 20 files. The HPL example references files numbered 0 to 19. The BASIC example references files named Ft0 through Ft19. Note that BASIC files can also be randomly accessed, unlike HPL tape files.

HPL	Translated Basic	Modified Basic
For I=0 to 19	FOR I=0 TO 19	FOR I=0 TO 19
ldf I,A$	ASSIGN #9 TO "File--I"	ASSIGN #9 TO "Ft" &val$(I)
	READ #9;A$	READ #9;A$
prt A$	PRINT A$	PRINT A$
Next I	NEXT I	NEXT I

In most cases converting to the file-by-name method simplifies the translated code and increases the readability of the translated program.

Using the file-by-name approach, the cartridge tape commands translate as follows:

fdf and trk Statements

The operations **fdf** and **trk** are no longer meaningful and are thus changed into a comment or assignment. A warning notes that the statement was deleted.

mrk and ert Statements

The statements **mrk** and **ert** are not translated. They can be replaced by a series of file create/file purge commands.

idf Statement

The operation **idf** does not translate. This operation is used to identify the contents of a file. The directory is not available from within a BASIC program.

Other Tape Operations

The following operations translate as shown and create a warning to indicate that a file name is needed:

- ldb → LOAD BIN "FILE--X"
- ldk → LOAD KEY "FILE--X"
- ldm → LOAD ALL "FILE--X"
- rck → STORE KEY "FILE--X"
- rcm → STORE ALL "FILE--X"

where "X" is the referenced file number.

rcf Statement

The **rcf** is translated into either a **SAVE** if it is a program save or to a **PRINT #9** sequence if it is a data save. If the **rcf** is a data store, the statement is translated to the following series of lines:

- ASSIGN #9 to "FILE--X"
- PRINT #9;Data List,END
- ASSIGN #9 to *

If the data save is a save of **r** variables, the statement is converted into a **PRINT** of the array **Rv**. In HPL, however, the data elements are saved in reverse order of how an array is saved. This should cause no problem for the translated program if the **r** variables are stored into a file and then read back into **r** variables. However, if the file is read back into simple variables or into an array, the order will be reversed.

If the **rcf** is a program store, the statement is translated as follows:

- SAVE "FILE--X"(,First line number(,Second line number))

If line numbers are contained in the **rcf** statement they are translated to the corresponding BASIC line number(s). A warning is generated that the line numbers were converted.

ldp Statement

The **ldp** is translated to a **GET** statement. The HPL line numbers, if any exist, are updated to the equivalent BASIC line number in this program. It must be remembered that the line numbers in this case may correspond to a different program, in which case the updated numbers will be wrong.

ldf Statement

The **ldf** statement translates into either a **LINK** if it is a program load or into a **READ #9;Data List** if it is a data load. In some cases, where the file number is not specified or where no data list follows, the syntax does not make it clear whether it is a program load or a data load. The 9825 solves the problem by reading the tape header. The Translator assumes that all questionable statements are program loads (nearly always correct). All **ldf** statements are flagged with a caution to check the translation.

If the statement is a data read, it translates to the following:

- ASSIGN #9 TO "FILE--X"
- READ #9;DATA LIST

If the statement reads **r** variables, it translates to a read of the array **Rv**. Note that if the file was stored from simple variables or arrays, the order will be reversed. If the load is from a file that was created by the Data File Translator from an **r** variable file, or the file was created in a BASIC program from an **Rv** variable store, the order will be correct.

In loading simple variables and arrays, the default conditions for HPL and BASIC in loading data files are different and not reconcilable. They are:

- If there are fewer data values in the tape file than variables in the data list, HPL loads the variables that can be loaded. BASIC generates an EOF or EOR error.
- If there are more data values in the tape file than variables in the data list, HPL generates an error. BASIC loads all the variables and continues.

If the **ldf** translates into a program load, a **LINK** is used. The HPL line numbers, if any, are translated, but the same cautions apply that applied to the **ldp** statement.

BASIC allows two different types of programs storage: as data files with GET, LINK, and SAVE statements and as program files with STORE and LOAD statements. All program statements are translated into GET, LINK, and SAVE statements which incorporate the same features as HPL. The complementary instructions STORE and LOAD execute much faster that the data file instructions but do not have the relocation capability nor a LINK capability. Linking must be done through COMmon storage similar to how the translator programs are linked.

Chapter 9
Translation Considerations — Option ROMS

String ROM

All statements implemented in the String ROM are implemented in BASIC and are translated.

The str Function

The **str** function in HPL and the **VAL$** in BASIC operate slightly different. In HPL, a positive expression outputs a leading blank that is not output in BASIC. A caution is issued. The translator outputs a space followed by a VAL$, making the positive value case correct. A negative value argument will have an extra space before the minus sign.

String Arrays

String arrays are represented differently in BASIC and HPL. A string array that is represented in HPL as A$[1] is represented in BASIC as A$(1). In order to properly convert strings, the string must be dimensioned within the program. The **INTRO** program allows you to enter any **dim** statements that might be necessary.

NOTE

If a string ARRAY is not dimensioned, it is not translated correctly. A string variable is.

NOTE

If either a string, string array, or array are not dimensioned in the program or dimensioned during **INTRO**, they cannot be entered into any COMmon statement used for Subprograms or Functions. An undimensioned string variable **A$** is entered into the COMmon statement as **A$[]**. An undimensioned array **A** is entered as **A()**.

Matrix ROM

All statements with this ROM are translated. The HPL instruction **inv** is broken down into two statements if the determinant of the matrix is desired.

Plotter ROM

The BASIC Plotter ROM supports communication with the 9872A Plotter (NOT the 9862A). HPL programs that run with the 9862A Plotter are converted to the appropriate BASIC commands, where possible, but will NOT operate with BASIC.

HP BASIC Plotter ROM Limitations

The BASIC Plotter ROM uses the AGL language for high level commands. The high-level commands differ much from the 9825 commands and therefore many statements are not translated directly.

Extended Plotter Statements Using Output Statements

Much of the plotter status that was stored in the plotter with the 9825 is now stored internal to the computer with the BASIC ROM. This means that changing the plotter status manually or with an extended plotter command through an **OUTPUT** statement may not have any effect on the plotter output. The **CSIZE** and **LABEL** are two examples.

It is not feasible, in general, to mix plotting modes in translated programs (i.e. translating 9825 plotter commands to BASIC plotter commands and 9825 **wrt** statements to **OUTPUT** statements because:

- The character size and shape may not agree.
- Plotter ROM statements in BASIC affect, in general, the internal state and not the physical plotter state. Thus, the pen, when moved manually or via an **OUTPUT** statement, will move back to where the computer last placed it prior to executing the next instruction.

The Translator does NOT resolve the conflicts that arise out of mixing plotter and extended plotter statements in the same program.

Axis Statements

The **axe** statement (9862A only) translates only if all parameters are present (intersection and tic-marks).

The 9872A **xax** and **yax** statements will have to implemented manually with the one BASIC axis instruction. To label the graph, a series of **LABEL** and **MOVE** commands will have to be used.

psc Statement

The **psc 0** statement translates to **PLOTTER Sc IS OFF**. The select code inserted is the last processed **psc** select code, or 705.

The **psc** statement is translated for the 9872A/HP-IB case. Thus, the statement:

```
psc A
```

translates to

```
PLOTTER IS INT(.01*A),A MOD 100,"9872A"
```

which is correct for the 9872A but not for the 9862A.

csiz Statement

The **csiz** statement is translated to **CSIZE**, but it produces no output to the plotter. The internal state of the computer is modified.

lbl Statement

Moving the pen prior to executing a **LABEL** has no effect as the pen jumps back to the position stored internally prior to plotting. The pen must be moved by a MOVE or PLOT statement prior to labeling.

In order to eliminate the CR and LF that is generated by the LABEL statement, an image of "#,K" is used. This will simulate very closely the 9825 **lbl** statement. When a positive number is printed on the 9825, a leading blank is output that is not output in BASIC (similar to the **str** function).

Other Non-Translatable Plotter Commands

The following statements do not translate directly:

- pclr – Plotter clear
- xax – X Axis-Will have to be combined with the **yax** into an AXIS statement.
- yax – Y Axis
- ofs – Origin offset-will have to be implemented with a new **SCALE** statement.
- cplt – Character position set. This can be implemented with a MOVE, but it depends on the aspect ratio, etc.

Flexible Disk ROM

Mass Storage Unit Specifier

The Mass Storage Unit Specifier is not added to the file names of flexible disk files. In most cases all transfers within a program will all be to either the cartridge tape or to the flexible disk. If the program contains flexible disk commands, then the following statement is added at the beginning of the program (unless a **drive** statement is the first flexible disk command encountered):

```
MASS STORAGE IS ":F8"
```

If both cartridge tape and flexible disk commands are used within the same program, one device will have to set up as the default mass storage device and the appropriate **msus** added to the file names of the other device.

cat Statement

A **cat** to a buffer or a string is NOT allowed and results in an error.

asgn and Assign Statements

The return variable returned from the HPL statement **asgn** and the BASIC statement **ASSIGN** do not match totally. A return of 0 or 1 is the same, and they are the primary values for which one is testing. In BASIC, a return value of 2 indicates that the file was either protected, the wrong file type, or was not available. HPL returns a value of 3 to 8 to indicate the status of the file that was not assigned.

ens Operator

The **ens** operator following a **sprt** or **rprt** does not have an equivalent in BASIC and is deleted. It is not possible, in BASIC, to replace a single line within a serial file. To replace a single line, the rest of the file will need to be rewritten.

type Function

The **type** function corresponds in value to the BASIC function **TYPE** except for the following:

	HPL	BASIC	Meaning
•	2.1	8	Starting portion of a string
•	2.2	9	Middle portion of a string
•	2.3	10	End portion of a string

BASIC also has several more return values that correspond to SHORT and INTEGER variables that are not used by HPL.

Other Non-Translated Statements

The following statements do not translate directly into BASIC:

- copy (Total disk copy)
- copy (Partial disk copy)
- dump (Disk dump)
- load (Disk load from tape)
- repk (Repack)

All statements associated with the system tape--those statements in the system tape, along with any binary keyword, are decompiled to **nxrom**.

General I/O ROM

Translation of fmt Statements to IMAGE Statements

The internal format of **fmt** and **IMAGE** statements are very different and require extensive decoding. In addition, the format of input and output **IMAGE** statements in the BASIC I/O ROM is different. This creates a potential problem if a format number is referenced by both an input and output statement. If that occurs, the format statement is flagged.

In HPL, a programmer has ten format statements that can be active at any one time. Program flow, not position in the program, determines which one is active. A BASIC program can have an unlimited number of **IMAGE** statements, but each input or output statement must reference a specific **IMAGE** statement. The translator keeps a cross-reference table of format numbers versus IMAGE statements. A format reference references the closest IMAGE statement directly above the reference, or, if only one **fmt** statement exists for a particular format number, it references that converted **IMAGE** statement regardless of its position in the program. If there is a potential problem with multiple format statements, the translator flags the format references. If the **fmt** statement is on the same HPL line as where it is referenced, it is not flagged. However, since that format statement might be used later on, it should be checked.

Some format specifiers cannot be completely translated. They are:

- An f, fz, or e statement that does not specify the number of decimal places reverts to the current fixed or float setting. Since that is not available in BASIC, a warning is generated and the number of decimal places is set to 0.
- The **c** specifier is replaced by the **K** specifier. A **c5** is also replaced by **K**.
- If the width of the field used with a f,fz, or e specifier is not specified, HPL left justifies the number. That is replaced by the BASIC **K** specifier, which does not allow a specific format (decimal places or exponential formatting).

Write Binary Statement wtb

The **wtb** is converted to the BASIC statement **WRITE BIN** if something is output and no strings or characters are output. Otherwise, it is converted to a **OUTPUT USING "#;---";---** statement. Thus, the statement:

wtb 7

translates to:

OUTPUT 7 USING "#"

and a complex statement such as:

wtb 7,A,32,C$,D

translates to:

OUTPUT 7 USING "#,B,B,K,B";A,32,C$,D

The translator translates all **wtb** statements for byte output. If words are to be output, then the specifier **W** or **Y** must replace the **B**.

Write Control Statement wtc

The **wtc** statement is replaced by a **WRITEIO** statement with a register number of 5.

Read Status (rds) and Read Interface (rdi) Functions

The Read Status and Read Interface functions are converted into statements in BASIC. Thus, an expression of the form:

```
rds(7)→B
```

is replaced by:

```
STATUS 7;B
```

However, if the statement is more complicated and cannot be directly replaced, the line is broken down into two BASIC lines. For example:

```
if rds(7);2→R          or  1+rds(7)→E
```

translates to:

```
STATUS 7;Temp          or  STATUS 7;Temp
IF Temp THEN R=2           E=1+Temp
```

The **STATUS** statement returns the I/O Interface card Status bit as bit 8 for the main return variable.

List Statement

In BASIC, the **LIST** is a command only and is not a statement that can appear in a program. Therefore, it is not translatable. The **list** can be simulated by **SAVE**ing the program in a data file and then printing out that data file. However, in many cases the ability to store the program as a data file eliminates the need for the ability to list from a program.

Extended I/O ROM

Octal/Decimal Operating Modes

BASIC does NOT contain an OCTAL operating mode for I/O. The **moct** and **mdec** statements are translated as follows:

	moct	mdec
to	mode=1	Omode=0

The Omode specifier is used in the utility function that simulates the **add** operator. Otherwise, octal mode is not translated. The translator generates a warning when the **moct** statement is translated.

Octal, Decimal, and Addition (add) Functions

The octal and decimal functions (**oct** and **dec**) are translated to the BASIC I/O ROM functions OCTAL and DECIMAL.

The binary **add** contained in HPL is translated to a function call. Included in the function call is the current operating mode (decimal or octal). If the program contains an **add** function, the utility function is automatically appended at the end of your program after translation.

Interface Control Statements

The following interface control statements and functions are translated into BASIC:

- wti — Write interface
- rdi — Read interface
- iof — I/O Flag
- ios — I/O Status

The **rdi** function is converted into a statement as was the **rds** function.

The **wti 0,<Select Code>** is translated to the statement:

Wti—sc=<Select Code>

Wti—sc is used for both the **wti** and **rdi** translations.

Table Conversion and Parity Statements

In BASIC, the **CONVERT** statement is intended to replace the **conv**, **ctbl**, and **par** HPL instructions. In addition, the **CONVERT** statement is capable of setting up a separate cross-reference table for each device. In HPL, the three commands are separate but apply to all devices. In converting the three HPL statements to BASIC, the following rules are followed:

conv Statement

When the **conv** is translated, a warning is issued that a select code is needed. Then it is broken down into as many separate **CONVERT** statements as are needed. The BASIC does not clear the convert table with each new **CONVERT** statement as does the 9825. The table must be manually cleared if desired.

ctbl Statement

When a **ctbl** is translated, a warning is issued that the select code needs to be entered. Then the statement is translated to a utility subprogram call to the subprogram **Ctbl** which is appended to the end of your program.

HPL starts the convert table with the character 0. BASIC starts the table with the character 1, with character 0 overlapping at location 256 in the input string. To correctly translate this statement, the entire 256 byte string must be generated by the utility subprogram. The translated **ctbl** statement will run much slower.

par Statement

When the parity statement is translated, it prints a warning that the select code is needed.

If a **ctbl** or **conv** statement is executed that will clear the parity, a warning will be issued that the parity may need to be reset.

Error Functions

The **rom** error function is not translated. The **ern** and **erl** functions are translated but are flagged, since the line number and/or error number returned will not match the error number and line number returned in HPL. Any routine that checks for a specific error number or error line will need to be updated to check for the corresponding BASIC error number and line number.

Device Names

HPL device names (**dev**) are translated to simple variables. The following statement:

dev "DVM",710

translates to:

Dvm—=710

Device names should be defined in the program being translated. If they are not defined in the program with a **dev** statement, they are still converted. However, because of the process of making a label, the BASIC label for that device name may not correspond to the name created in another program (although it is not likely that will happen). If the device names are unique (i.e. there are no names with the same letters but capitalized differently), there will be no problems.

HPL Command equates (**equ**) convert to string variables in a similar manner. They must be defined in the translated program to be interpreted correctly.

Note that, for the **dev**, **equ**, and **buf** statements, the symbols are converted in the same manner (truncation, capitalization, etc) as HPL line labels. Also, the strings used for command equates are not dimensioned but are implicitly dimensioned to 18 characters by BASIC. If the string is longer than 18 characters, it must be dimensioned by use of a **DIM** statement.

Buffered I/O

BASIC does not contain the buffer statements or a transfer statement. However, each transfer in BASIC is buffered and each input or output statement allows you to define the transfer mode (i.e. DMA, Fast Handshake with bytes or words, etc.). Thus, much of the power of the 9825 buffered I/O is built into the BASIC I/O structure.

The buffered I/O characteristics are partially simulated as described here:

- All buffers are translated to string variables. If a **buf** statement equates a buffer to a string variable, then that statement has the effect of substituting the string name for the buffer name.

- All reads and writes from/to a buffer are translated to output statements from/to the value area as described in the I/O Manual. Any outputs to a buffer have the effect of writing to that entire string. Any inputs from a buffer have the effect of reading from that entire string. Any input or output statement is flagged when the destination or source is a buffer.

- No buffer pointers are kept. The statement:

  rds ("buffer")

  translates to:

  LEN(Buffer—$)

  which only simulates partially the operation of the buffer pointer.

- The **tfr** statement is not translated. The transfer operation could be replaced by the suitable **ENTER** or **OUTPUT** statement using the various modes allowed by the computer I/O ROM, or each individual read or write to the buffer could be changed.

Timeout Statement

The HPL timeout statement **time** is translated to **SET TIMEOUT**. A warning is generated noting that the select code and also the timeout linkage are needed. The timeout operates differently in HPL and BASIC. In HPL, the timeout creates an error that is trapped by the **on err** statement. BASIC creates an interrupt when a timeout occurs. Thus, the code to process a timeout must be transfered from an error routine to an interrupt routine. Interrupts must also be enabled.

Interrupts

Abortive interrupts are not allowed in BASIC. If an abortive byte exists, the byte is deleted, and a warning is generated.

Interrupts will likely appear different in BASIC due to the different operating speed. Also, more end-of-lines occur in BASIC because multiple statements per line are not allowed.

HP-IB Programming

Multiple Bus Addressing

The Extended I/O ROM of the 9825 allows multiple devices to be addressed on either input or output statements. Input statements are translated to two statements, a **CONFIGURE** statement followed by the **ENTER** statement. **OUTPUT** statements are directly translated.

Other HP-IB Statements

All other statements involving HP-IB communication are translated.

The **trg** statement asserts the attention line but does not bring it back down. An additional statement might be needed after the trigger, such as a **STATUS**, to bring the line back down.

The I/O ROM for BASIC is needed for all General and Extended I/O ROM statements of the 9825 as well as for HP-IB transfer.

Advanced Programming ROM

For/Next Loops

The FOR/NEXT loops translate directly into BASIC. The **by** is replaced by **STEP**.

Subprograms and Functions

Subprograms and functions for HPL are translated into subprograms and functions in BASIC. A maximum of 100 subprograms are allowed. The subprograms in BASIC are vastly different in the following ways, however:

- All HPL variables are global. Variables in the BASIC routines are local. Therefore, all variables used must be placed into COMmon storage.
- BASIC does not allow jumps outside the subprogram environment. HPL does.
- HPL allows subprograms to exist anywhere. Subprograms in BASIC must be at the end of the program.
- HPL considers format references as global. BASIC requires the IMAGE to be within the subprogram.

From the above list, it is clear that there are many differences between HPL and BASIC subprograms.

Changing Subprograms to Subroutines

In many cases in HPL a **cll** is used instead of a **gsb** because the **cll** is executed immediately and the **gsb** is executed at the end of the line. HPL lines can be conserved using this technique. During the translation process separate BASIC lines are formed anyway. Therefore, a more efficient translation will result if all unnecessary **cll**s are changed to **gsb**s. Subprograms that are good candidates to change to simple subroutines are:

- Routines that have no passed parameters.
- Routines that use no local **p** variables.
- Routines that are not called from another routine that will be translated as a subprogram or function.

Even some subprograms that use local **p** variables or passed parameters can be more efficiently translated to a normal subroutine.

Subprogram and function calls are executed much slower in BASIC than in HPL. If speed is critical, they should be changed to subroutines. Functions can be changed also to subroutines, although the modification required is greater than for a subprogram.

Since HPL subprograms are identified by their **cll** references, it is necessary to change any unnecessary **cll**s to **gsb**s. Either of these two methods can be used: Before starting translation, change the selected **cll** to a **gsb**. This may require creating new HPL lines if the **cll** is on the same line as a **gto** or **gsb**. For example, the line:

1: cll **subprog**;gto "end"

would have to change to:

1: gsb "subprog" 2: gto "end"

Select the interactive mode. Then, when **PASS1B** has been loaded and before the actual subprogram translation begins, you will be given the name of each subprogram and asked if it should remain a subprogram or be changed to a simple subroutine. If you desire it changed to a subroutine, all **cll**s referencing that subprogram will be changed to **GOSUB**s.

Creation of Common Statements for Subprograms

HPL considers every variable to be global. A subprogram or function can use a variable defined in the main program. In BASIC, all variables are considered local within the subprogram unless passed or in COMmon. While this is an advantage for structured programming and modularity, it creates difficulty in translating HPL subprograms to BASIC. To aid you, **PASS1B** does the following:

- All variables used within the subprograms and functions are put in a COMmon statement. An appropriate COMmon statement is put at the start of the program and also at the start of each subprogram, if necessary.
- If you, during the **INTRO** routine, input a COMmon statement(s) as part of the file linkage, that COMmon statement is used as the basis for building the COMmon statements in the present program, thus simplifying the linking of programs.

Determining Subprogram and Function Limits

HPL subprogram endings may be very difficult to determine. To determine where a subprogram ends, the following rules are applied:

- Each **gto** or **gsb** is cataloged and the destination line tagged as belonging to the present environment.
- Potential terminators are:

  ret
  end
  ldf
  ldp
  get
  chain
  gto

  Whenever one of the above terminators is encountered, the program looks ahead to see if any future lines are tagged with the present environment. If so, the subprogram or function is continued. If not, it is terminated.

- If a new subprogram entry point is encountered and an old subprogram has not been terminated, one of three things will happen:

  1. If the operator has selected Interactive Mode, the translator will beep twice and ask you to indicate whether the present subprogram should be terminated.

2. If the Interactive Mode is not active and the two subprogram entry points are adjacent, the present subprogram is continued. The second entry point is flagged, and the calls to that entry point are flagged as invalid calls.

3. If the Interactive Mode is not active and the two entry points are not adjacent, then the first subprogram is flagged and terminated and the second subprogram begun. A linkage will need to be manually created from the first subprogram to the second.

- If a line tagged as belonging to the main program is encountered within a subprogram, the subprogram is terminated. If the Interactive Mode has been selected, you will be asked to define how the translation should continue.
- If the subprogram contains an untranslated **jmp** statement, the subprogram is not terminated until a line is found that is flagged with another environment (such as a line starting another subprogram). In the Interactive Mode you are allowed to define the end of the subprogram if the program flow is not clear.

Adjustment of Subprogram and Function Calls

HPL allows calls to a subprogram or function to have a variable number of parameters. The translator determines the maximum number of parameters passed and makes all calls be that length. Zeros (00) are passed if necessary.

If a subprogram uses **p0**, which is set to the number of parameters passed, the number of passed parameters is added to the pass parameter list of each call to that subprogram, and the variable **P0** is added to the subprogram header.

Relocation of Subprogram and Functions

BASIC subprograms and functions must exist at the end of a program. In HPL they can exist anywhere and may not even be contiguous. The translator relocates contiguous subprograms and functions (i.e. subprograms whose code is all in a straight line) to the end of the program.

Identifying Subprograms and Functions

In HPL, subprograms and functions are identified by their calls. During **PASS1** all calls to subprograms and functions are tabulated. If a subprogram is not called, it is not translated as a subprogram.

Relocation of Comment Lines

Comment lines (stand alone labels) and blank lines just before the starting line of a subprogram or function are relocated to within the subprogram body.

Subroutines within a Subprogram

Subroutines within a subprogram are not detected, although **gsb** statements within a subprogram are flagged. Any **ret** within that subroutine is changed to **SUBEND** or **SUBEXIT** and will have to be converted back.

p VARIABLES

p variables are converted to the BASIC variables **Pv0**, **Pv1**,etc. If a reference is made to a **p** variable in the form **pp0** or **pN**, the variable is converted to an array form similar to the **r** variables. You will need to equate that array variable to the passed **p** variables if necessary.

Other Non-Global HPL Parameters

Several of the HPL parameters are global in HPL but will not be global in the BASIC subprograms. Among them are IMAGE statments (an IMAGE must be in the same environment as the subprogram), device names, and equate strings. IMAGE statements will need to be duplicated in the subprogram. Equate names and buffer names can be passed or placed into COMmon. The translator will detect device names and place them into common storage. If a COMmon statement was input during the **INTRO** routine containing a device name, that device name is equated within the program to the current device name being processed only if all letters of the names are the same so that a one-to-one correspondence between the BASIC name and the HPL name can be determined.

Several other parameters are reset when entering a BASIC subprogram. The trigonometric mode is reset to RADIANS.

Optimizing the Translation of Subprograms and Functions

To optimize the translation effectiveness of subprograms and functions, the following should be done:

- Make sure, either by modifying the HPL source prior to translation or by using the interactive mode as explained, that no subroutines are needlessly called as subprograms.
- If the subprograms and functions involve a large number of variables in COMmon, the COMmon statement can be optimized to reduce the length of the COMmon statement contained in each subprogram.

Split and Integer Precision Storage Functions

The functions **fts** (Full to Split), **stf** (Split to Full), **fti** (Full to Integer), and **itf** (Integer to Full) are converted to BASIC function calls that call the utility routines contained in **ADD-FN**. Those utility routines are appended at the end of the translated program if needed by the translator. BASIC offers a variety of ways to store variables (such as INTEGER, SHORT) that HPL does not offer except through these functions. The translated program will run much faster if the storage functions are converted to numeric expressions using the "INTEGER" or "REAL" storage forms. The **fti** and **itf** can also be converted to single line functions with an increase in the execution speed.

Cross Reference Statement

The **xref** statement is not translated. The utility library contains a BASIC program that will perform a cross-reference on a program stored as a data file (**SAVE**d).

Systems Programming ROM

Intelligent Terminal Instructions

The **on key** and **kret** statements are translated. The **key** statement is translated to the **KBD$** function, although the action of the two is different. The HPL **key** function returns only the next character in the buffer while the **KDB$** returns the entire buffer.

The functions **asc** and **bred** are not translated.

End-of-Line Specification

The **eol** function is translated to the BASIC I/O ROM statement **EOL**. However, for the end-of-line specification to have effect, it must be referenced in the IMAGE statement with the **L** specifier.

Serial Interface Control Statements

The following commands are translated:

- wsc to WCONTROL
- wsm to WMODE
- rss to RSTATUS

The remainder of the Systems Programming statements are not translated, with the exception of the free-syntax operator. They are:

- asc
- bred
- rkbd
- store
- nal
- avm
- cln

H-P Basic System Enhancements

The enhanced BASIC language makes programming much easier. The ability to indent lines allows a program to be segmented to aid understanding. Use of multiple-character names also allows the program to be better documented.

BASIC offers the unified mass storage concept in storing data on tape or on disk. This allows the program to run with either mass storage device without program modification.

BASIC offers a variety of ways that data can be stored internally and on tape. Common storage for subprograms, integer, and short precision are all allowed.

The plotting capability has been greatly enhanced with BASIC, although the translation from HPL to BASIC is not direct. There is a much larger set of instructions for use in plotting.

BASIC automatically buffers all I/O. Also, each file for the mass storage device can be buffered. This can be used to advantage, especially in the **OVERLAP** mode, to speed up the processing of an I/O or Mass Storage intensive program.

Appendix A

Error Messages and Explanations

The Translator prints out WARNINGS and CAUTIONS within the BASIC listing where difficulty was encountered. In general, a WARNING means that the statement was not completely or accurately translated and will need to be edited manually. A CAUTION means that the statement was translated, but it needs to be examined for possible difficulty. Below is a list of the translator CAUTIONS and WARNINGS and their meanings:

Translator Cautions

CAUTION ABORTIVE BYTE DELETED

Abortive Interrupts are not allowed in BASIC. Only end of line interrupts exist. Therefore, the abortive byte was deleted.

CAUTION CALL CONVERTED TO GOSUB

Using the Interactive Mode, a Subprogram is now being translated as a simple subroutine. This CALL is being changed to a GOSUB.

CAUTION CHECK TRANSLATION

The following statement belongs to the **ldf** or **cmd** class where the exact intent of the statement is determined by the 9825 at run time. The statement may not have been translated correctly. Several other instructions are flagged with this warning. Subtle differences in the way BASIC and HPL operate may require modification of the line or following lines.

CAUTION CONVERT MODE ASSUMED OUTPUT

The **conv** statement is assumed to apply to the output direction.

CAUTION ens DELETED

The **ens** at the end of the following statement was deleted.

CAUTION ERROR FUNCTION

The line contains either a **erl** or **ern** function which, although they are translated, may need checking in their use.

CAUTION FUNCTION NOT ALLOWED

A function call is not allowed at this position in BASIC (usually following a **prt** or **wrt**). The function call will have to be moved to a preceding line and a temporary variable defined.

CAUTION gsb IN SUBPROGRAM

A **gsb** was detected in this subprogram/function. The **ret** associated with that inbedded subroutine may have been changed to **SUBEXIT** or **SUBEND**.

CAUTION HPL LINE NUMBERS BEING TRANSLATED

This program load or store command contains line number references which have been translated to the corresponding BASIC line number for this program. That may not be correct for the application and thus should be checked.

CAUTION ILLEGAL SUBPROGRAM ENTRY POINT

This line is called by a **cll** or a function call but was not translated into a subprogram entry point. In BASIC, the only point that a subprogram/function can be entered is through the first line. The statements that call this line will be flagged as invalid calls.

CAUTION IMAGE ERROR AT LOCATION

The **fmt** statement had an item that was not recognized at the specified character position in the line.

CAUTION **K** SUBSTITUTED FOR

The BASIC image format specifier **K** was substituted for the indicated HPL format specifier. This is done when the width is not specified (as if **f.2**) and the result should be left justified. Note that the **K** outputs the number in standard format.

CAUTION MULTIPLE DEFINED LABEL

The line label has already been defined previously.

CAUTION MAXIMUM r VARIABLE USED UNDETERMINED

It may not have been possible, during PASS 1, to determine the maximum **r** variable used, and thus the dimensioning of the array **Rv** may not be correct.

CAUTION MULTIPLE DEFINED FORMAT REFERENCE

This **red** or **wrt** statement references a format number for which there is more than one corresponding IMAGE statements. You should, using the cross-reference tables, verify that the right IMAGE statement is referenced.

CAUTION not OPERATOR

The following line contains the operator **not**. Since the mathematical priority of the **not** operator is different in HPL and BASIC, the line should be inspected to see if any parenthesis are needed.

CAUTION NULL PROMPT

The following **ent** or **enp** contains a null prompt. In HPL this means that any previously displayed comment will remain on the screen for the variable prompt. To operate correctly, a ; will need to be inserted after the **DISP** statement that generates the desired prompt. This will require looking at the program flow to determine which **DISP** should be changed.

CAUTION PARITY MAY BE NEEDED

This statement resets any parity condition that might be set, and thus the parity might need to be reset.

CAUTION PLOT CONTROL OF 0

A plot control parameter of 0 does not have the same effect in BASIC.

CAUTION Rd MISSING IN

The Rd parameter is missing in the **fmt** statment parameter listed. HPL takes the default setting of the **fxd** or **flt** statement for the decimal positions. The translated version will have 0 decimal places.

CAUTION REFERENCE TO INPUT OR ERROR FLAG

Either Flag 13 or Flag 15 is referenced. Neither Flag has a counterpart in BASIC.

CAUTION RETURN VALUE MAY BE DIFFERENT

The value returned by the corresponding BASIC function may be different from that returned by the HPL function. (i.e. **asgn** vs. **ASSIGN**)

CAUTION rnd SEED DELETED

The **rnd** seed was deleted from the **rnd** function and was negative. If you want to initialize the seed, then use the **RANDOMIZE** statement.

CAUTION SE OR DB AFTER rcf DELETED

The debug and secure flags were stripped from the program record command.

CAUTION str FUNCTION

The **str** function was translated into the BASIC **VAL$**. However, if the function expression is positive, **str** outputs a leading blank that is not output in BASIC. Therefore, the length of the string returned in HPL is one character longer than that returned in BASIC. An extra space is inserted by the translator which may make the negative condition incorrect.

CAUTION SUBPROGRAM ENVIRONMENT PROBLEM

This HPL line is reference by a previous HPL line in another subprogram/function or main program segment.

Translator Warnings

WARNING ENVIRONMENT BRANCH ERROR

This line branches to a line outside the current program segment (i.e. outside the current subprogram or function or into a subprogram or function if in the main program)

WARNING eol REQUIRES IMAGE REFERENCE

For the following **eol** sequence to have effect in BASIC, if must be referenced in an IMAGE statment with the **L** operator.

WARNING FORMAT USED FOR BOTH INPUT AND OUTPUT

The format number of this **fmt** statement is referenced by both **wrt** (output) and **red** (input) statements. In BASIC there are slight differences in an input and output image statement. This image statement has been translated for output. You should check and make sure the translation is correct.

WARNING FILE NAME NEEDED TO REPLACE FILE

A file name is needed to replace the dummy file name **FILE — X** that was inserted by the translator. The referenced file number is given.

WARNING FORMAT NOT FOUND

The format number referenced was not found in the current program segment being translated.

WARNING I/O TO BUFFER

The following statement does input or output to a buffer. Since I/O to a buffer will not work the same in BASIC as it does in HPL, the line should be reworked.

WARNING ILLEGAL BUFFER TYPE

The buffer type specified in the **buf** statement was not 0,1,2,3, or 4.

WARNING INPUT f FORMAT ERROR

The specified format element is invalid for an input **IMAGE** statement.

WARNING INPUT EXP FORMAT AT

The specified format element is invalid for an input **IMAGE** statement.

WARNING INTERRUPT LINK TO STRING

The **on int** statement set up a link to a string. BASIC requires the link to be to a line number or label.

WARNING INVALID CALL

This is either a **CALL** or function call that calls a line that was not translated as a Subprogram/Function entry point, or it was a call that was converted to a **GOSUB** through the interactive mode and passed parameters.

WARNINGINVALID dim

One of the arrays was dimensioned using another variable or array. BASIC requires that all arrays in the main program be dimensioned with integers.

WARNING jmp NOT TRANSLATED

The **jmp** instruction was not a simple integer jump. It will have to be hand translated and replaced with a BASIC computed **GOTO** or computed **GOSUB**.

WARNING LABEL CANNOT BE MADE UNIQUE

The mentioned label could not be made unique.

WARNING moct NOT VALID MODE

The BASIC language does not allow for a decimal and octal operating modes. Decimal is the only valid mode.

WARNING OPERAND DELETED

The stated operand was deleted and made into a comment.

WARNING OPERAND NOT TRANSLATED

An operand in the following line was not translated at all, or the translation is not correct for BASIC. You will have to hand translate that instruction.

WARNING p VARIABLE ERROR

A p variable in the following line was accessed as an array and thus was left in that form. All p variables are passed as simple variables. You will either need to change the line or else define the elements of the array **Pv** from the various passed p variables.

WARNING PASS2 SYNTAX PROBLEM

The PASS2 routine encountered a syntax that caused a fatal error. That error was trapped and printed and the translation continued.

WARNING SEGMENT ILLEGALLY TERMINATED

The above program segment was not properly terminated before a new program segment was begun. Generally this means that there was a **gto** or **gsb** to a line that has not been encountered, but the next line either starts a new subprogram/function or is jumped to from the main program.

WARNING SELECT CODE NEEDED

The statement requires a select code in BASIC to operate. Replace the **Sc** with the desired select code.

WARNING STRING VARIABLE — MIS-DIMENSIONED

The listed string variable was either never dimensioned in the translated program, or was dimensioned more than once. This problem can be cured in linked programs by supplying the dimensioning information before the translation begins (during the **INTRO** routine).

WARNING SYNTAX ERROR

The following line contains a syntax problem that caused a fatal error during either PASS1 or PASS1B. The fatal error was trapped, printed, and the translation continued.

WARNING TIMEOUT LINKAGE NEEDED

The timeout linkage is different in BASIC and HPL. in HPL, the **time** statement creates an error and can be trapped by an **on err** statement. In BASIC, the **SET TIMEOUT** creates an interrupt, and thus an interrupt routine must be set up.

Translator Fatal Errors and Remedies

Listed below is a summary of the problems that could cause the translator to halt. In most cases the problem is caused by an array being too small and overflowing, or by the input or output file being too small in size. The arrays used in the translator are large and should be sufficient for the vast majority of programs, but in order to fit the program into 64K of memory the size had to be constrained. Listed are the potential problems and their solutions:

1. Error 59 in Program segment **TAPE**:
   The input file is too small — re-create with a larger size and start over.
2. Error 59 in Program segment **PASS1**:
   The destination file is too small — re-create with a larger size and start over.
3. Error 59 in Program segment **PASS1B**:
   The buffer file used by **PASS1B** was too small. If the file **SRC1** existed, then it was used as the buffer file. Otherwise, the INBUF file was used. Re-create with a larger size and start over.
4. Error 59 in Program segment **PASS1B**:
   The destination file, when attempting to add the subprograms, was too small. Re-create with a larger size and start over.
5. Error message — **Label$ array overflow-translator fails**
   The string variable **Label$** has overflowed. Either the program being translated must be segmented into two smaller parts, or the Translator can be re-run from the **INTRO** routine and the following line executed from the live keyboard after the verbage has been printed:

   Label—limit=X

   where X is some number less than 16. This truncates the long labels as they are compiled to X characters. If the program was nearly through the first pass when the problem occured, then 14 should be ok. If a large number of HPL lines were left to be translated when the error occured, then 12 or 10 should be used.
6. Error 17 in Program Segment **PASS1**:
   The line number cross-reference table **Line—xref** overflowed. The HPL program contained more than 750 lines. To correct, the program must be segmented and run in two parts.
7. Error 17 in Program segment **PASS1B**:
   The program to be translated contains more than 100 subprograms or functions. The program will have to be segmented to allow translation.

Subject Index

a

add 44
ADD-FN 5,53
Advanced Programming ROM 48
Arrays 32
Arrays, String 37
asc 54
asgn 40
ASSIGN 40
avm 54
axe 39

b

bred 54
Buffered I/O 46
by 48

c

cat 40
CAUTIONS 55
chain 50
cll 49
cln 54
COM Statement 9
CONFIGURE 48
conv 45
CONVERT 45
Convert 45
copy 41
cplt 40
csiz 39
CSIZE 38,39
ctbl 45

d

Decimal 44
DEG 27
deg Mode 27
Device Names 46
dim 19,20
DIM Statement 9
Dimensioning 8
dump 41

e

end 50
ens 41
ENTER 48
EOL 53
erl 45
ern 45
Error Messages 23,24
Error Remedies 61
Errors, Fatal 61
ert 33
Extended I/O ROM 44

f

Fatal Errors 61
fdf 33
FIXED 0 27
Flexible Disk ROM 40
fmt 41
For/Next 48
fti 53
fts 53
Functions 18,48,52
Functions, Maximum 26
fxd 2 27

g

General I/O ROM 41
get 50
GOSUB 49
gsb 49,50
gto 50

h

HP-IB 10,14,48
HP-IB Interface 14
HPL Flags 31
HPL Lines, Maximum 26

i

idf 33
IMAGE 41,42
Imbeddings 31
INBUF 4,6,7,10,25
Integer 53
Interactive Mode 8
Interrupts 47
INTRO . 3,4,5,10,13,14,19,20,26,37,50,52
INTROD 5
iof 44
ios 44
itf 53

j

jmp 30

k

KBD$ 53
KEY 10

l

LABEL 38,39
Label Characters, Maximum 26
Label-limit 29
lbl 39
ldb 33
ldf 19,34,50
ldk 33
ldm 33
ldp 34,50
LEN 47
Line Numbers 8
LINK 18,35
Linked Programs 17
List 43
LISTER 4,11,29
LOAD 18
load 41

m

MAKETP 3,5
Matrix ROM 38
max 32
mdec 44
Messages 21
Messages, Error 23,24
Messages, Non-Error 22
min 32
moct 44
MOVE 39
mrk 33

n

nal 54
not 30

o

Octal 44
ofs 40
OUTBUF 4,7,25
OUTPUT 48
OVERLAP 54

p

p Variables 52
par 45
Parity 45
PASS1 4,10,13
PASS1B 4,10,49,50
PASS2 4,10,11
pclr 40
Plotter ROM 38
Prompts 21
prt 27
psc 39
Pv0 52

r

r Variables 30
RAD 27
rcf 19,34
rck 33
rcm 33
rdi 43,44
rds 43,47
READIO 4
Real 53
Remedies, Error 61
repk 41
Requirements 3
ret 50
rkbd 54
ROM:
- Advanced Programming 48
- Extended I/O 44
- Flexible Disk 40
- General I/O 41
- Matrix 38
- Plotter 38
- String 37
- Systems Programming 53

rprt 41
rss 53
RSTATUS 53

s

SEARCH 28
SEARPL 28
Segmented Programs 17
SET TIMEOUT 47
Special Function Keys 19
sprt 41
SRC1 10,25
STATUS 43
STEP 48
stf 53
store 54
STORE PROGRAM 15
str 37
String Arrays 37
String ROM 37
Subprograms 18,48,52
Subprograms, Maximum 26
Systems Programming ROM 53

t

TAPE 4,6,10,13
Tape Commands 32
TCMPL 4
Time 47
Timeout 47
tlist 19
trg 48
trk 33
type 41
TYPE 41

u

units 30

v

VAL$ 37

w

Warnings 58
WCONTROL 53
WMODE 53
WRITE BIN 42
WRITEIO 43
wrt 38
wsc 53
wsm 53
wtb 42
wtc 43
wti 44

x

xax 40
xref 53
XREFAL 28

y

yax 40

Part No. 09835-10031
Rev. B

Printed in U.S.A.
September 2, 1979

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 63 ROCKAWAY, N. J.

Postage will be paid by addressee

HEWLETT-PACKARD
Green Pond Road
Rockaway, N.J. 07866

Attn. Digital Product Manager

SOFTWARE REGISTRATION

To receive all information on future modifications or improvements to this library, please complete this registration form and return it to Hewlett-Packard.

(Please Print)

Library Part Number _____ Revision _____ Date Received _____

Comments _____

_____

_____

Name _____ Title _____

Company _____ Tel. _____

Address _____

City _____ State _____ (Province) Zip _____ (Code)

Country _____

Software Registration Card

MADE IN U.S.A.

HP PART NO. 9282-0563